
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**
**DEPARTMENT OF INFORMATION SYSTEMS**


# WEBOVÁ APLIKACE PRO VIRTUÁLNÍ TRENÉRSTVÍ

**WEB APPLICATION FOR VIRTUAL TRAINING**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **JAROSLAV VYSTAVĚL**
**AUTHOR**

**VEDOUCÍ PRÁCE** **Doc. Ing. JAROSLAV ZENDULKA, CSc.**
**SUPERVISOR**



**BRNO 2017**

## Abstrakt

Tato práce se zabývá popisem vývoje webové a mobilní aplikace. Cílem webové aplikace je tvorba fitness plánů na základě vstupních informací od uživatelů. Cílem mobilní aplikace je notifikovat uživatele na skutečnosti vyplývající z fitness plánu. Obsahem této technické zprávy je popis požadavků na aplikace, její návrh a popis implementace. Práce byla implementována na klientské straně pomocí technologií HTML, CSS a JavaScript s využitím rámce CanvasJS. Serverová strana se skládá z PHP skriptů. Data jsou uložena v databázi MySQL. Mobilní aplikace je postavena na platformě Android.



## Abstract

This thesis is concerned with the development of web application and mobile application. The main goal of the web application is creating a fitness plan based on the information obtained from users. The goal of the mobile app is to alert users of fitness plan-related notices. The content of this technical report is the description of the application requirements, its design and description of the implementation. The work was implemented on the client side using HTML, CSS and JavaScript technologies using the CanvasJS framework. The server side consists of PHP scripts. The data is stored in the MySQL database. The mobile app is built on Android platform.



## Klíčová slova

webová aplikace, mobilní aplikace, Android, PHP, Java, fitness



## Keywords

web application, mobile application, Android, PHP, Java, fitness



## Citace

VYSTAVĚL, Jaroslav. *Webová aplikace pro virtuální trenérství*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Jaroslav Zendulka, CSc.

# Webová aplikace pro virtuální trenérství

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. Ing. Jaroslava Zendulky, CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jaroslav Vystavěl
18. května 2017

## Poděkování

Děkuji za cenné rady, ochotu, čas a především perfektní přístup ze strany Doc. Ing. Jaroslava Zendulky , CSc., který mi poskytl odborné vedení. Velice oceňuji jeho práci. Dík patří i Ing. Vojtěchu Frömlovi, který mi pomáhal v počátcích této práce s koncepcí zadání.

*Zadání bakalářské práce/20006/2016/xvysta02*

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav informačních systémů Akademický rok 2016/2017

# Zadání bakalářské práce

Řešitel: **Vystavěl Jaroslav**
Obor: Informační technologie
Téma: **Webová aplikace pro virtuální trenérství**
**Web Application for Virtual Training**
Kategorie: Informační systémy

Pokyny:

1. Seznamte se principy tvorby webových aplikací.
2. Analyzujte požadavky na webovou aplikaci pro automatické vytváření dietních a tréninkových plánů dle specifických požadavků. Aplikace dále umožní sledovat postup fitness plánu, porovnání s dosaženými výsledky a prostřednictvím notifikací bude uživatele upozorňovat na události v souvislosti s tréninkovým plánem.
3. Navrhněte webovou aplikaci splňující výše uvedené požadavky. Využijte k tomu vhodné modelovací techniky.
4. Webovou aplikaci implementujte a ověřte její funkčnost.
5. Zhodnoťte dosažené výsledky a další možné pokračování v tomto projektu.

Literatura:

- Stackeová, D.: Fitness programy - teorie a praxe: metodika cvičení ve fitness centrech. Galén, 2008
- Zakas, N.Z.: JavaScript pro webové vývojáře - Programujeme profesionálně. Computer Press, 2009
- Naramore, E. et al.: PHP5, MySQL, Apache: Vytváříme webové aplikace, Computer Press, 2009

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zendulka Jaroslav, doc. Ing., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
*vedoucí ústavu*

# Obsah

# 1 Úvod

Dnešní doba je charakteristická moderními technologiemi vyskytující se na každém kroku. Chytrá zařízení pronikají i do oblastí, kde bychom si je před několika málo lety dovedly jen stěží představit.

Dalším trendem, který se stal nesmírně populární je aktivní životní styl. Lidé vždy sportovali, ale nyní je tento způsob života ještě více zpopularizován. Do jisté míry za tento vývoj mohou sociálně sítě, kde běžný smrtelník může prostřednictvím příspěvků sledovat, co jeho (sportovní) idol denně dělá a je tímto způsobem života částečně ovlivňován.

Takový motivovaný jedinec se rozhodne, že by se chtěl fyzicky přiblížit svým vzorům a začne navštěvovat fitness centrum. Z mé vlastní zkušenosti vím, že začátky jsou těžké. Velká část začátečníků poprvé přijde do posilovny a neví co dělat, neví jak se používá který stroj, neví jak správně provádět cviky, tak aby si neublížili. Právě pro tyto účely by měl být v každém fitness centru zaměstnán trenér, který by měl umět ve všech ohledech poradit začínajícím cvičencům. Mnohé posilovny toto splňují, problémem ale může být vysoká vytíženost klienty, kdy se ze strany trenéra zkrátka nedostane na všechny. Výsledkem toho stavu nezřídka bývá to, že klient už do posilovny nevkročí.

Tomuto scénáři se dá předcházet vytvořením jednoduché webové aplikace, jejíž výstupem může být nejen vytvoření fitness plánu, ale také popis konkrétních cviků. Díky podpoře responzivního designu snadno zobrazitelném i na menší uhlopříčce mobilního zařízení, by bylo snadné vzít si svého osobního „virtuálního trenéra“ sebou do fitness centra a cvičit podle vytvořeného plánu.

Správný vyznavač fitness životního stylu ví, že samotné cvičení je jen část úspěchu. Další neméně důležitá věc je otázka stravování. Reálný trenér by uměl poradit v obou oblastech. Webová aplikace by to s určitou dávkou abstrakce měla rovněž zvládnout a umět vytvořit i jídelní plán.

Webová aplikace pro virtuální trenérství tedy vytváří jakýsi průnik trendy dnešní doby – mobilní telefon a aktivní životní styl.

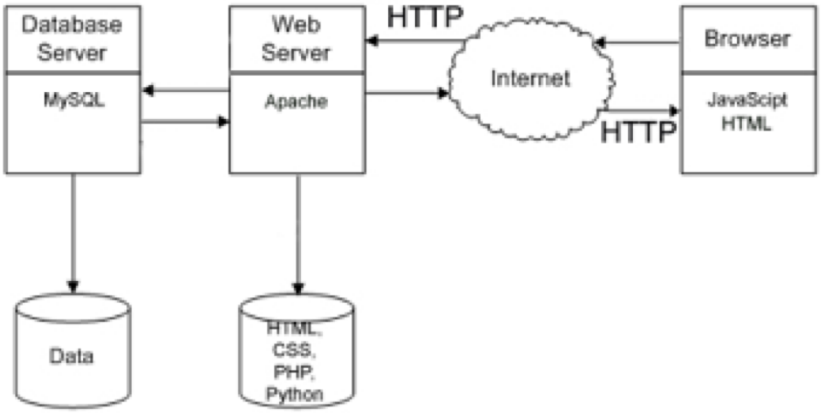
Tato práce dokumentuje formou technické zprávy postupy, které jsem uplatnil při vývoji aplikace. Text práce je členěn do 4 kapitol. Za úvodem následuje obecný popis praktik při tvorbě webových stránek a aplikací, včetně popisu vybraných technologií. Dále se práce skládá z analýzy požadavků kladených na nároky aplikace, kde je vysvětlena problematika fitness, navržený ER diagram a model případů použití. Kapitola Návrh řešení a implementace popisuje způsob jakým jsem postupoval při výběru platforem a technologií, nástin implementace některých složitějších případů užití. V kapitole testování je popsán průběh testování a její implementační dopady na aplikace. Následuje shrnutí dosažené práce v kapitole závěr a diskuze o jejím možném využití.

# 2 Principy tvorby webových aplikací

V této kapitole je vysvětleno, co je chápáno pod pojmem webová aplikace. Jsou konstatovány její výhody a nevýhody v porovnání s ostatním typem aplikací a je nastíněn současný stav na webu.

## 2.1 Webová aplikace

Jedná se o aplikaci, která je fyzicky uložena na webovém serveru a uživatel přistupuje k jejím datům jako klient. K tomuto účelu používá internetový prohlížeč (*tenký klient*) zpracovávající data v podobě značkovacího jazyka HTML, která převádí do vizuální podoby. Na Obr. 2.1 je znázorněna architektura webové aplikace využívající databázový server. Komunikace mezi server a klientem probíhá HTTP protokolem, ten funguje na principu *dotaz-odpověď*, klient (webový prohlížeč) si vyžádá obsah, ten, v případě, že existuje a klient k němu má právo, je odeslán současně s kódem *200 OK*, značícím úspěšné vyřízení požadavku. Důležitým poznatkem je, že server sám o sobě vůči klientovi žádnou aktivitu nevyvíjí. Komunikaci tedy iniciuje vždy klient. Server, na kterém může být spuštěný webový i PHP server, může spolupracovat s databázovým serverem. Z obrázku je patrné, že prostřednictvím HTTP si obě strany zasílají HTML obsah. Hlavním prostředkem pro provádění operací na klientské straně bývá JavaScript. PHP se odehrává pouze na serverové straně.




*Obr. 2.1 Architektura webové aplikace*

Samotné webové stránky jsou tvořeny zejména HTML obsahem upraveným kaskádovými styly (CSS) využívanými v mnoha podobách a frameworcích pro formátování. Stránky můžeme dělit na dynamické a statické. Statická webová stránka se většinou skládá z kombinace HTML a CSS a je uživateli doručena v takové podobě, v jaké je uložena. Struktura těchto stránek vypadá tak, že každá má svůj vlastní soubor s příponou *.html nebo *.php. Úvodní strana, kterou uživatel spatří po otevření webu, se nazývá index.html.

Dynamické stránky jsou většinou generovány za běhu. Důležitým prvkem je možnost interakce uživatele s jejím obsahem. K tomuto účelu lze využít skriptování. Důsledkem, většinou, bývá vizuální změna obsahu, či operace s databází.

Může se zdát, že statický web je již překonaný dynamickými technologiemi, ale i v dnešní době se najdou účely, pro které je vhodný (prezentace firem, organizací...).

Nezpochybnitelnou výhodou webové aplikace oproti jiným typům je její snadná aplikovatelnost ve smyslu, že uživatel nemusí nic instalovat (za předpokladu, že již má nainstalován prohlížeč, případně zásuvné moduly, které jsou využívány). Paměťové nároky jsou tedy minimální. Dalším plusem, který plyne již ze samotné podstaty, je nezávislost na platformě klientského zařízení. Nezáleží na tom, zda

uživatel přistupuje na stránku aplikace z mobilního zařízení na bázi Androidu, iOS nebo například ze stolního počítače s OS Linux. Přístupem z kteréhokoliv z těchto přístrojů by měl uživatel docílit stejných výsledků. Jediný vizuální či funkční rozdíl napříč různými zařízeními, který by mohl vzniknout souvisí s verzí prohlížeče a podporou zásuvných modulů přítomných na klientské straně. Otázka vzhledu webové stránky do jisté míry souvisí i s *responzivním designem* (pozn. responzivní design - schopnost webu přizpůsobit svůj vzhled i pro řádově menší displeje mobilních zařízení), který by měl být prioritou každého vývojáře. Jako poslední výhodu je třeba zmínit jednoduchost aktualizace. Z pohledu klienta stačí pouze znovu načíst stránku a prohlížeč ze serveru stáhne nová data v aktualizované podobě. Pro vývojáře to znamená pouze provést aktualizaci dat na serveru. Nelze tedy počítat se scénářem, že uživatel používá neaktuální verzi.

Seznam nevýhod je kratší a částečně souvisí s obsahem předchozího odstavce. Značným problémem může být nutnost připojení k Internetu. V případě, že uživatel využívá internetové připojení poskytované mobilním operátorem, je tato výhoda ještě o něco významnější (z důvodů datového limitu). Nicméně poměr mezi velikostí datového limitu (zejména v dnešní době, kdy se tato hodnota velmi často pohybuje v rozmezí několika GB) a množstvím stažených dat při přístupu a využití služby je poměrně vyhovující ve prospěch aplikace.

Co se týče zabezpečení, je nutné vyžadovat po uživatelích autentizace (přihlášení do systému jako celku) ve složitějších aplikací obsahujících například citlivé osobní údaje i autorizaci (ověření, že uživatel má právo vstoupit).

## 2.2 Technologie používané pro vývoj na straně klienta

Tato podkapitola obsahuje popis technologií užívaných ke tvorbě webových aplikací na serverové a klientské straně. V současnosti je využívané velké množství platforem a jejich frameworků. Rozhodl jsem se, dle svého uvážení, vybrat ty nejvíce významné a popsat jejich podstatu a základní vlastnosti.

### 2.2.1 HTML

První verze HTML (*HyperText Markup Language*) se objevila v roce 1990 v CERNu. Autory jsou Tim Berners-Lee a Robert Cailliau.

HTML dokument se od obyčejného textového souboru liší tím, že zahrnuje kromě vlastního obsahu stránky navíc i informace o formátování stránky, které do textu nepatří a jsou určeny pouze pro prohlížeč. Tyto informace jsou předávány pomocí značek uzavřených mezi znaky `< >`, např. `<html>`. K většině značek existuje jejich protějšek, který ukončuje jejich platnost (párové značky), takové značky se vyznačují lomítkem před jejím názvem (`</html>`).

#### Struktura HTML dokumentu

Kromě značek je třeba aby měl každý HTML dokument pevně stanovenou strukturu. Celý zdrojový text je umístěn mezi párové značky `<html>` a `</html>`. V hlavičce ohraničené značkami `<head>` a `</head>` se nacházejí například informace o kódování, definice stylů nebo zobrazovaný název stránky.

```html
<html>
    <head>
        <title>První stránka</title>
        <meta name = "description" content = "první stránka v HTML">
        <meta name = "keywords" content = "ukázka, stránka, škola">
        <meta name = "author" content="Jaroslav Vystavěl">
        <meta http-equiv="content-type" content="text/html; charset=windows-1250">

    </head>
    <body>
        <p>Ahoj Světe!</p>
    </body>
</html>
```

*Obr. 2.2 Možná struktura HTML dokumentu*

V hlavičce se nacházejí i tzv. metainformace, tyto se vkládají pomocí značky `<meta>`. Obsah této značky se řídí dle informace přiřazené do obsahu atributu `name`. Na Obr. 2.2 lze vidět popis, klíčová slova a jméno autora.

## Možnosti HTML

HTML umožňuje efektivně zobrazovat rozličné informace. Umožňuje formátovat text i bez nutnosti použít CSS, ačkoliv se tato technika pro složitější weby téměř vůbec nepoužívá, je to možné. Velmi používané značky jsou `<div>` a `<span>`. Soustředí obsah do bloků, kterým lze snadno nastavit nějakou společnou vlastnost. Velice jednoduchá a efektivní je tvorba odrážkových a číslovaných seznamů. Ukázka je na Obr. 2.3.

```html
<h4>Seznam uvnitř seznamu:</h4>
<ul>
  <li>A</li>
  <li>B
    <ul>
      <li>b1</li>
      <li>b2</li>
    </ul>
  </li>
  <li>C</li>
</ul>
```

**Seznam uvnitř seznamu:**

- A
- B
  - b1
  - b2
- C

*Obr. 2.3 Vnořený seznam uvnitř jiného seznamu (vlevo kód, vpravo výsledek)*

K dalším základním značkám patří `<table>`, určená pro tabulky. Tato značka má poněkud složitější strukturu, ale o to lepší výsledek nabízí. Řádky tabulky jsou tvořeny pomocí `<tr>`, jednotlivé buňky pomocí `<td>`. Nadpis sloupce lze vytvořit jako `<th>název</th>`. Samotnou tabulku lze rozdělit značkami `<thead>`, `<tbody>` a `<tfoot>` na záhlaví, ve kterém jsou specifikovány nadpisy, tělo tabulky a zápatí. Všechny zmíněné značky jsou párové. Praktická ukázka je na Obr. 2.4. Barevného odlišení jednotlivých součástí bylo dosaženo pomocí CSS.

```html
<table>
  <thead>
    <tr>
      <th>A</th>
      <th>Hodnota</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>Výsledek</td>
      <td>3</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>A1</td>
      <td>1</td>
    </tr>
    <tr>
      <td>A2</td>
      <td>2</td>
    </tr>
  </tbody>
```

| A | Hodnota |
|---|---|
| Výsledek | 3 |
| A1 | 1 |
| A2 | 2 |

*Obr. 2.4 Demonstrace značek pro tvorbu tabulek*

### HTML5

V aktuální verzi, HTML5, došlo k několika změnám. Je možné vynechat značky `<html>`, `<head>` i `<body>` a výsledný dokument bude i přesto validní. K dalším změnám oproti předchozí verzi HTML4 uvedené v roce 1997 patří[6]:

- podpora off-line módu, kdy je po povolení této vlastnosti pomocí atributu ve značce `<html>` možné pracovat s lokálním úložištěm dat,
- podpora přehrávání multimediálního obsahu přímo v prohlížeči,
- zjednodušený zápis `doctype` na začátku souboru,
- množství nových značek, zejména pro strukturování stránek,
- nové typy formulářových polí
- a další...

## 2.2.2 CSS

Původní verze Cascading Style Sheets (CSS) vznikla v roce 1996 [5]. Jedná se o jazyk sloužící pro úpravu a formátování HTML dokumentu. Pomocí HTML tedy určujeme *co* má být obsahem, pomocí CSS upravujeme *jak* má obsah vypadat po vizuální stránce.

Styl CSS je možné do stránky zahrnout třemi způsoby:

1. Přímým zápisem – k jednotlivým HTML značkám (*tagům*) pomocí atributu `style`. Tento způsob se ovšem používá velmi zřídka z důvodu nepraktičnosti.
2. Značka `<style>` – do hlavičky HTML dokumentu lze přidat párovou značku `<style>` a dovnitř napsat specifikaci stylu jednotlivých elementů
3. Externí soubor – pomocí zápisu na Obr. 2.5. V atributu `href` je uvedena cesta k souboru se styly.

```html
<link rel="stylesheet" href="styl.css">
```

*Obr. 2.5 Externí stylopis CSS*

### 2.2.3 JavaScript

Autorem JavaScriptu (původní název byl LiveScript) je Brendan Eich, který v době vývoje pracoval pro Netscape. Tento skriptovací jazyk vznikl v roce 1995 na popud uživatelů. Bylo velmi nepříjemné, např. vyplnit formulář, čekat třicet sekund až se stránka odešle a požadavek zpracuje a poté obdržet zprávu o tom, že jste zapomněli vyplnit textové pole. Bylo by velice výhodné provádět tuto kontrolu na straně klienta[4].

JavaScript je interpretovaný, objektový, skriptovací jazyk, který je multiplatformní. Ve stránce většinou plní úlohu různých dynamických prvků jako je tvorba menu roletek, formátování textu, výše zmíněné ověřování formulářů, atd.

Problémem při využití na webu může být jeho zákaz ze strany uživatele, či nekompatibilita s verzi internetového prohlížeče.

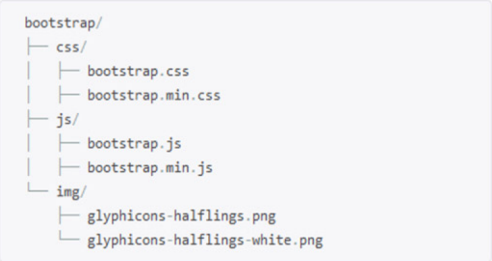
Pro podporu práce s JavaScriptem existuje množství frameworků jejichž cílem je zefektivnit a zjednodušit jeho použití. Odlišují se od sebe rozdílnou funkcionalitou a zaměřením. Za všechny lze jmenovat *Node.js,* umožňující spouštět skripty JavaScriptu netypicky na straně serveru. *AngularJS* umožňuje, v podstatě nedává na výběr, vytvářet aplikace za použití návrhového vzoru MVC (jedná se o návrhový vzor, který rozděluje datový model aplikace, uživatelské rozhraní a její logiku). Z hojně užívaných knihoven lze zmínit např. *jQuery*, která usnadňuje práci s DOMem (Document Object Model). Pro tvorbu grafů lze využít například Chart.js nebo *CanvasJS*.

### 2.2.4 Bootstrap

S HTML, CSS a JavaScriptem úzce souvisí další produkt používaný při tvorbě webu. Jedná se o front-end framework postavený na bázi těchto technologií. Jeho obrovská výhoda tkví v řadě předpřipravených nástrojů a tříd, které uživatel v kódu pouze používá. Odpadá tedy např. definice vlastních komponent (tříd, identifikátorů) v CSS.

Pomocí Bootstrapu lze relativně snadno vytvořit vzhledem pokročilý web, jehož použití by mělo být pro uživatele příjemné a intuitivní. Šetří práci a čas kodérů pomocí předpřipravených šablon.

Šablony jsou znovupoužitelné a lze je je libovolně modifikovat úpravou příslušného `.css` souboru.




*Obr. 2.6 Souborová struktura Bootstrapu*[7]

Na obrázku 2.1 je patrná souborová struktura tohoto frameworku. V každé složce jsou uloženy soubory jednoho typu.

Za zmínku stojí `.min` verze stejně pojmenovaných souborů stejných typů. Je to jejich tzv. minimalizovaná verze, ze které byly odstraněny bílé znaky pro snížení velikosti. Rozsah nezmenšeného souboru bývá poměrně rozsáhlý, protože většinou obsahuje stovky definic stylů a pravidel. Je tedy výhodné je tímto způsobem zmenšit.

Vývojář má možnost do těchto definic libovolně zasahovat a přizpůsobovat si je svým účelům. Nicméně jako osvědčený postup bych doporučil vytvořit si nový `.css` soubor, redefinovat pravidla a tímto přepsat stávající. Výhodou této metody je, že při aktualizaci na novou verzi Bootstrapu není nutné manuálně přepisovat vše, co jsem pracně měnil.

Pro použití responzivního vzhledu je nutné vložit veškerý obsah mezi koncové značky tagu `<div>`.

Původ Bootstrapu sahá do roku 2010, kdy byl vyvinut Markem Ottem a Jacobem Thorntonem, původně, pro interní účely Twitteru [9]. V srpnu 2011 byl jeho kód vložen na GitHub volně ke stažení a již o 6 měsíců později se stal nejpopulárnějším open-source projektem na tomto repozitáři.[8]

## 2.3 Technologie používané pro vývoj na straně serveru

### 2.3.1 PHP

PHP (původně *Personal Home Page*, nyní známá rekurzivní zkratka *PHP: Hypertext Preprocessor*) je skriptovací jazyk jehož počátky se datují k roku 1995, kdy dánský programátor Rasmus Lerdorf vyvinul skript, který fungoval jako počítadlo přístupů jeho stránek. Tento skript dělal dvě věci: protokoloval informace o návštěvníkovi a zobrazoval počet návštěv.[2]

PHP je nejčastěji použito pro vývoj webových aplikace a obecně webových stránek. Nicméně jeho uplatnění lze najít i při tvorbě konzolových aplikací. Oblíbenost PHP pramení z jeho relativní jednoduchosti k naučení a také z velké palety vestavěných funkcí připravených pro použití vývojářem. Na Obr. 2.7 je ukázka oblíbeného programu *Hello World*.

Podrobněji se tomuto tématu budu věnovat v kapitole Návrh řešení a implementace.

```
1.  <?php echo '<p>Hello World</p>'; ?>
```

*Obr. 2.7 Ukázka PHP kódu*

### 2.3.2 Python

Vznik Pythonu je vázán ke městu Amsterodam a roku 1989. Jeho autorem je Guido van Rossum, který v roce 1991 vydal první verzi jazyka [28]. Autor si jako cíle při jeho tvorbě stanovil to, že má jít o jazyk intuitivní a jednoduchý pro naučení, ale zároveň mocný, aby byl schopný se prosadit mezi konkurenty. Další cíl si zvolil to, že by mělo jít o jazyk, který by šlo číst jako běžnou mluvu (angličtinu) a také otevřenost kódu.
Ukázka syntaxe je uvedena na Obr. 2.8. Další fakta jsou zmíněna v kapitole Návrh řešení a implementace.

```
1.  print("Hello world!")
```

*Obr. 2.8 Oblíbený Hello world program v Python verze 3.*

### 2.3.3 MySQL

Jedná se o multiplatformní systém řízení báze dat, který umožňuje technologiím PHP a Apache kooperovat na zpřístupnění a zobrazení dat ve formátu čitelném v internetovém prohlížeči. Vzhledem k rozšíření mezi uživateli lze najít podporu u drtivé většiny hostingů.
Tento relační databázový server byl poprvé vydán pro veřejnost v roce 1995 firmou TCX DataKonsult[2].

Jednoduchý příklad SQL dotazu lze najít na Obr. 2.9. Další informace jsou uvedeny v kapitole Návrh řešení a implementace.

```
SELECT * FROM Zakaznici;
```

*Obr. 2.9 Ukázka jednoduchého SQL dotazu. Předpokladem je existence tabulky Zakaznici*

# 3 Analýza požadavků

Obsahem této kapitoly je popis požadavků na aplikaci. V kapitole 3.1 je nastíněna problematika týkající se fitness životního stylu nutná k pochopení této práce. Následující kapitola 3.2 obsahuje neformální popis požadavků a očekávané funkcionality. Diagram případů užití se nachází v kapitole 3.2.1 včetně strukturovaného popisu dvou vybraných případů užití a dalších uvedených nestrukturovaně.

## 3.1 Úvod do problematiky fitness

Vzhledem k tomu, že problematika není všeobecně známá, považuji za nutné vysvětlit základní pojmy z oblasti fitness včetně systému tréninku a skladby jídelního plánu. Pokud je čtenář cvičencem, dovoluji si tvrdit, že může tuto podkapitolu přeskočit a pokračovat na 3.2. V následujícím textu budu, vzhledem ke kontextu, používat termín *cvičenec*, namísto uživatel.

V první řadě je nutné si uvědomit rozdíl mezi *tréninkovým* plánem a *dietním* (nebo také jídelním) plánem.

### Tréninkový plán

*Tréninkový* plán sestává ze cviků, které cvičenec vykonává např. ve fitness centru, nebo kdekoliv jinde za předpokladu vlastnictví potřebného vybavení. Obsah plánu lze dělit na menší segmenty. Pro ilustraci uvádím příklad, který podrobně popíšu.

| Den | Partie | Cvik | ZS | PS | | Op. |
|---|---|---|---|---|---|---|
| Pondělí | Hrudník | bench-press | 3 | 4 | | 6 |
| Pondělí | Hrudník | tlaky s jednoručkami | 1 | 3 | | 10 |
| Pondělí | Hrudník | peck-deck | - | 3 | | 12 |
| Pondělí | Biceps | zdvih s velkou osou | 2 | 3 | | 8 |
| Pondělí | Biceps | EZ činka na Scott. | - | 3 | | 10 |
| Pondělí | Biceps | kladivové zdvihy | 10 | 4 | | 7 |
| Pondělí | Břicho | leh-sedy | - | 4 | | 15 |
| Středa | Záda | Přítahy (shyby) | 2 | 5 | | 8 |
| Středa | Záda | přítahy velké osy | 1 | 3 | | 7 |
| Středa | Záda | mrtvý tah | 1 | 3 | X | 10 |
| Středa | Triceps | bench-press na úzko | 2 | 3 | | 8 |
| Středa | Triceps | francouzský tlak | - | 4 | | 6 |
| Středa | Triceps | stahování kladky | 3 | 3 | | 15 |
| Pátek | Nohy | dřepy | 3 | 3 | | 6 |
| Pátek | Nohy | leg-press | 2 | 4 | | 8 |
| Pátek | Nohy | zakopávání | - | 3 | | 12 |
| Pátek | Ramena | tlaky s velkou činkou | 2 | 4 | | 8 |
| Pátek | Ramena | rozpažování | - | 4 | | 10 |
| Pátek | Ramena | přítahy osy k bradě | 2 | 4 | | 12 |
| Pátek | Lýtka | výpony na lýtka | - | 8 | | max. |

*Obr. 3.1 Příklad tréninkového plánu*

Na Obr. 3.1 lze vidět konkrétní plán. Tento plán je sestaven autorem práce. Pro lepší orientaci je použito barevné zvýraznění korespondujících částí.

Plán je tedy rozdělen na den, tělesnou partii, cvik, zahřívací série (ZS), pracovní série(PS) a opakování v sérii (Op.).

- Den, tzv. *tréninkový den* – je uveden v plánu proto, že cvičenec by měl v daný den vyvíjet tréninkovou aktivitu.
- Partie, *tělesná partie* – cvičenec každý trénink (1 den = 1 trénink) cvičí několik, zvykem bývají max. 3, tělesné partie. Vyšší počet nemá pro začátečníka, vzhledem k jeho předpokládaným fyzickým a silovým dispozicím, smysl.
- *Cvik* – procvičení jedné partie spočívá v provedení několika cviků.
- *Zahřívací série* – ze zdravotních důvodů je vhodné se zvláště na začátku série procvičit, aby nedošlo k poškození svalů a šlach. Počet opakování v těchto sériích není blíže specifikován a je ponechán na uvážení.
- *Pracovní série* – zde by měl cvičenec vykonávat co největší úsilí, pro největší nárůst svalové hmoty.
- *Opakování* – jedná se o počet provedení jednotlivých cviků v jedné sérii. Obvykle se značí písmenem „X“, tedy symbolem „krát“. Př. *bench-press: 4x6*, znamená že v pracovní sérii je provedeno celkem 24 opakování rozdělených do 4 bloků (*sérií*).

## Jídelní plán

*Jídelní plán* je své podstatě seznam potravin, které by měl uživatel zkonzumovat během dne. Potraviny dělíme na základě primárních zdrojů živin, esenciálních pro svalový růst, na bílkoviny a sacharidy. Následuje krátká odbočka do oboru sportovní výživy.

- *Bílkoviny* (proteiny) – jedná se o látky složené z aminokyselin, které jsou základním stavebním kamenem svalů; z hlediska funkčního jsou základem pro enzymy, hormony a imunitní systém [10].
- *Sacharidy* – chemicky se jedná o hydráty uhlíku; říká se jim také cukry, zjednodušeně řečeno, jsou palivem pro lidské tělo [10].
- *Tuky* – estery mastných kyselin a glycerolu; otázka ideálního přijímaného množství je neustále diskutovaná, nicméně tuky plní v lidském těle nenahraditelnou funkci, např. při vstřebávání vitamínů a tvorbě hormonů [10].

Obr. 3.2 obsahuje příklad dietního plánu pro 80kg sportovce. Tento konkrétní obsahuje i rozepsané údaje o tom, kolik která potravina obsahuje své primární živiny. Tuky je v tomto případě možné zanedbat z důvodu vysoké pravděpodobnosti, že cvičenec asi není vrcholový sportovec, který by musel řešit i to, kolik gramů tuků denně přijme.

Jak již napovídá předchozí odstavec, je tedy nutné soustředit se na objem přijatých bílkovin a sacharidů. Doporučené množství bývá obvykle stanoveno v rozmezí 1-2g/tělesný kg u bílkovin a 3-4g/tělesný kg u sacharidů.

Samotné pořadí potravin není nutné striktně dodržovat.

| Čas | Potravina | Množství (g) | Bílkoviny (g) | Sacharidy (g) |
|---|---|---|---|---|
| 7:00 | ovesné vločky s mlékem | 100 | - | 68 |
| 7:00 | proteinový koktejl | 30 | 25 | - |
| 9:30 | cottage sýr | 150 | 17 | - |
| 9:30 | tmavé pečivo | 80 | - | 60 |
| 12:00 | kuřecí maso | 180 | 45 | - |
| 12:00 | rýže | 120 | - | 40 |
| 12:00 | zelenina | 80 | - | - |
| 14:00 | celozrnné pečivo | 80 | - | 60 |
| 14:00 | tvrdý sýr | 100 | 25 | - |
| 16:00 | (trénink) | - | - | - |
| 18:00 | sacharidový nápoj | 120 | - | 80 |
| 19:00 | losos | 150 | 30 | - |
| 19:00 | brambory | 200 | - | 30 |
| 22:00 | tvaroh | 250 | 27 | - |
| 22:00 | celozrnné pečivo | 80 | - | 60 |
| | Celkem | | 169 | 398 |

*Obr. 3.2 Příklad jídelního plánu*

## 3.2 Požadavky na aplikaci

Aplikace by měla splňovat několik cílů. Hlavní myšlenkou, jak je již z názvu patrné, je vytvoření dietního a/nebo tréninkového *individuálního* plánu na základě vstupních informací předaných uživatelem ve formě vyplněného formuláře.

Funkční požadavky lze rozdělit na základě role uživatele – neregistrovaný uživatel, registrovaný uživatel a administrátor.

*Neregistrovanému uživateli* je nabídnuta pouze výše zmíněná základní funkcionalita: tvorba dietního plánu, tréninkového plánu nebo obojí. Pochopitelně se může také zaregistrovat.

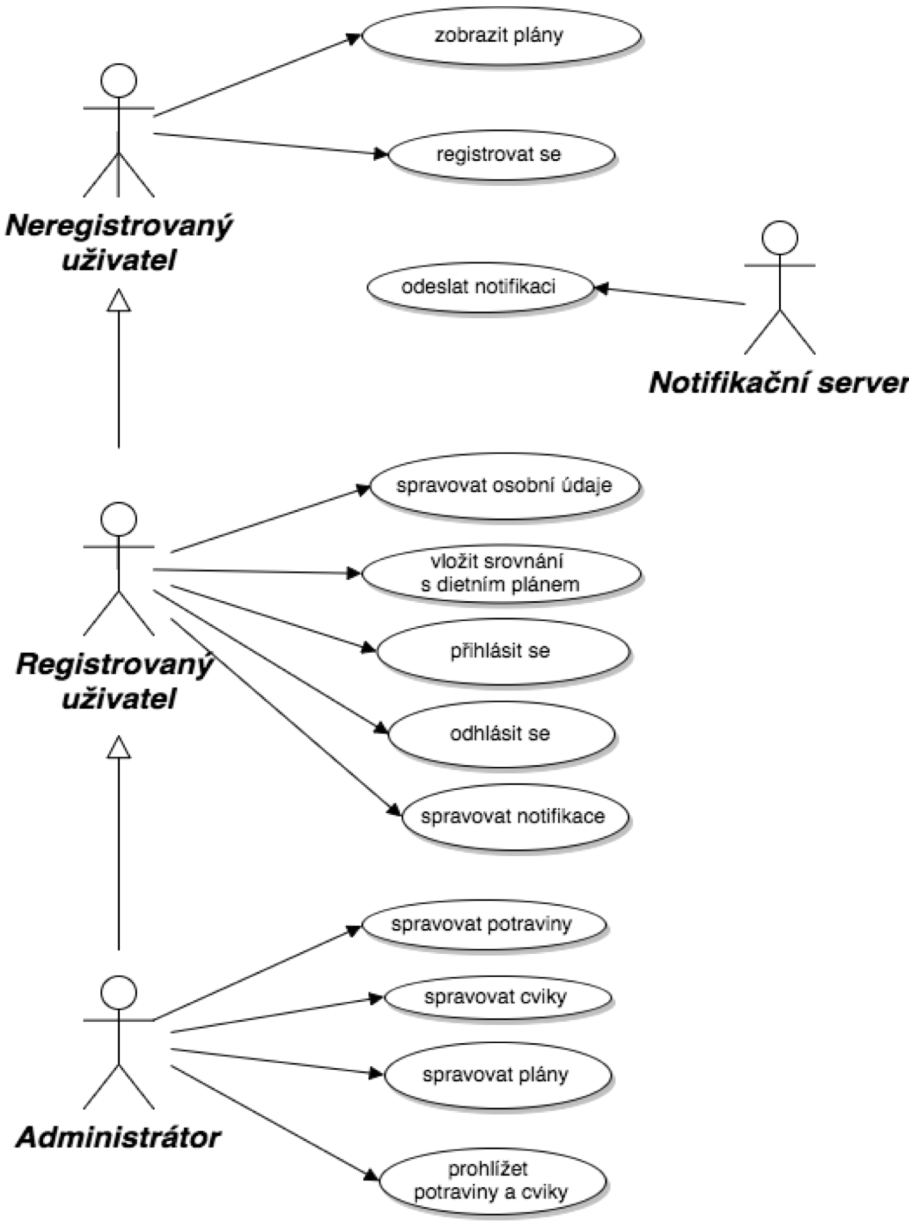
Pokud se uživatel *zaregistruje*, po přihlášení získá několik funkcí navíc. Mezi tyto funkce patří například: sledování změn hmotnosti a stavu podkožního tuku, porovnání jídelního plánu a možnost notifikace obsahu plánu na mobilní telefon.

Dále je nutné mít v systému i uživatele s právy administrátora. Za tímto typem uživatele si lze typicky představit trenéra fitness centra, který rozumí dané problematice a je schopen korektně sestavovat plány na základě požadavků. Tento uživatel by měl mít pravomoci úpravy obou typů plánů, vkládání nových potravin a cviků a jejich úpravy včetně smazání.

### 3.2.1 Model případů použití

Návrh složitějšího počítačového programu se neobejde bez důkladného návrhu zachycujícího požadavky na aplikaci, funkcionalitu a její chování při interakci s uživateli. Pro tyto účely vznikl diagram případů užití (anglicky use case diagram). Diagram zobrazuje chování systému *z pohledu cíle případu použití pro uživatele*. Diagram je na Obr. 3.3.

Dále následují jednotlivé případy užití. První dva jsou uvedeny ve strukturované podobě, zbylé jsou popsány nestrukturovaně.




*Obr. 3.3 Diagram případů užití*

| **Případ užití: spravovat plány** |
|---|
| Účastníci:<br>• administrátor (*primární aktér*) |
| Vstupní podmínky:<br>1. Administrátor je úspěšně přihlášen do systému |
| Tok událostí:<br>1. Administrátor zvolí volbu Upravit tréninkový plán<br>2. Administrátor vybere plán, který chce upravit<br>3. Aplikace vyhledá plán a zobrazí okno pro úpravy<br>4. Nově vytvořenou položku vyplní informacemi (den, zahřív. série, prac. série, opakováni) dle svého uvážení |

| |
|---|
| 5. Administrátor potvrdí změny<br>6. Aplikace uloží upravené informace do databáze<br>7. Aplikace zobrazí okno s výsledkem |
| Následné podmínky:<br>1. Záznamy odpovídající tréninkovému plánu s odpovídajícím číslem jsou upraveny a uloženy |
| Rozšíření:<br>4a: Administrátor chce nejdříve odstranit položky<br>4a1: Administrátor vybere položky k odstranění<br>4a2: Administrátor potvrdí změny<br>4a3: Aplikace znovu zobrazí uživateli okno s úpravami<br>1b: Administrátor nejdříve vytvoří nový cvik, který je vložen do plánu<br>1b1: Administrátor vybere cvik<br>1b2: Administrátor vyplní položky pro vložení cviku<br>1b3: KDYŽ došlo k chybě při vyplňování formuláře, opakuje pokus |

| **Případ užití: spravovat notifikace** |
|---|
| Účastníci:<br>• registrovaný uživatel (*primární aktér*)<br>• notifikační server |
| Vstupní podmínky:<br>1. Uživatel je přihlášen<br>2. Uživatel má vyplněny vstupní informace a je mu přiřazen dietní plán |
| Tok událostí:<br>1. Uživatel si stáhne aplikaci do svého mobilního zařízení na platformě Android<br>2. Uživatel se na mobilním zařízení přihlásí do svého účtu pod stejným uživ. jménem a heslem jako do web. rozhraní<br>3. KDYŽ uživatel zadá nevalidní vstupní informace, opakuje pokus<br>4. Vstupní informace jsou zpracovány<br>5. Uživateli je na e-mail odesláno ID jeho zařízení notifikačním serverem, které vloží v aplikaci do vyžadovaného textového pole<br>6. Aplikace uloží dvojici uživ. jméno - ID zařízení<br>7. Uživatel v tabulce plánu vybere zaškrtnutím zaškrtávacího políčka položky, o kterých chce být notifikován a nastaví čas<br>8. Serveru je odeslán požadavek na zasílání notifikace, tyto jsou naplánovány |
| Následné podmínky:<br>1. Uživatel má úspěšně nastavený systém notifikací |

*Neregistrovaný uživatel* má možnosti omezené pouze na dva případy užití. Volba **zobrazit plány** spočívá ve vyplnění formuláře o jeho tělesných vlastnostech, na základě kterých, je mu z databáze

přiřazen konkrétní plán. Pokud uživatel vyžaduje zpřístupnění dalších funkcí aplikaci, může se **zaregistrovat** vložením e-mailové adresy a zvolením hesla. Tímto vznikne v databázi nový uživatel.

Po **přihlášení** uživatelský jménem a heslem, zadaným v předchozím kroku, plní roli *registrovaného uživatele*, který má větší paletu možností. Ihned po prvním přihlášení by měl **vyplnit osobní údaje**, které jsou následně uloženy do databáze pod jeho uživatelským jménem. Pod touto volbou se také rozumí údaje pro zhodnocení postupu ve fitness plánu – hodnota podkožního tuku a tělesná hmotnost. K oběma údajům je nutné přiřadit datum, kdy byla stanovena. Aplikace tyto údaje uloží do databáze. **Vyplnit srovnání s dietním plánem** znamená, že je z databáze načten dietní plán přiřazený uživateli. Ten poté vyplní, které položky zkonzumoval a na základě těchto údajů a těch uložených v databázi je vypočteno procentuální splnění očekávaného množství sacharidů a bílkovin. Celá agenda notifikací je zahrnuta v případu užití **spravovat notifikace**, který je uveden ve strukturované podobě. Může se samozřejmě také **odhlásit** ze systému.

*Administrátor* má možnost **spravovat potraviny**, tedy vkládat nové potraviny do databáze a také je mazat. To stejné platí o cvicích – **spravovat cviky**. Pod možností **spravovat plány** je chápán proces úpravy dietního i tréninkové plánu. V případě dietního plánu, který je zvolen na základě vybraného čísla, je vyobrazena aktuální podoba plánu v tabulce. Administrátor může měnit pořadí jednotlivých položek, mazat je a měnit jejich obsah. Obsahem se rozumí název potraviny a její množství.

U tréninkového plánu je scénář obdobný: výběr čísla plánu, zobrazení tabulky s plánem a jeho úprava. Je možné měnit, ve kterém dni bude cvik cvičen, odebrat jej, upravovat počty jeho sérií (pracovních i zahřívacích), počty opakování a samozřejmě také cvik jako takový. Uživatel-administrátor má možnost **prohlížet potraviny a cviky**, v případě potravin si může zobrazit nutriční informace (obsah bílkovin, sacharidů, tuku, energetickou hodnotu) a typ primárního zdroje (bílkoviny/sacharidy). U cviků uvidí jejich textový popis, vhodnost při problémech s páteří a partii, které náleží.

### 3.2.2 Datový model

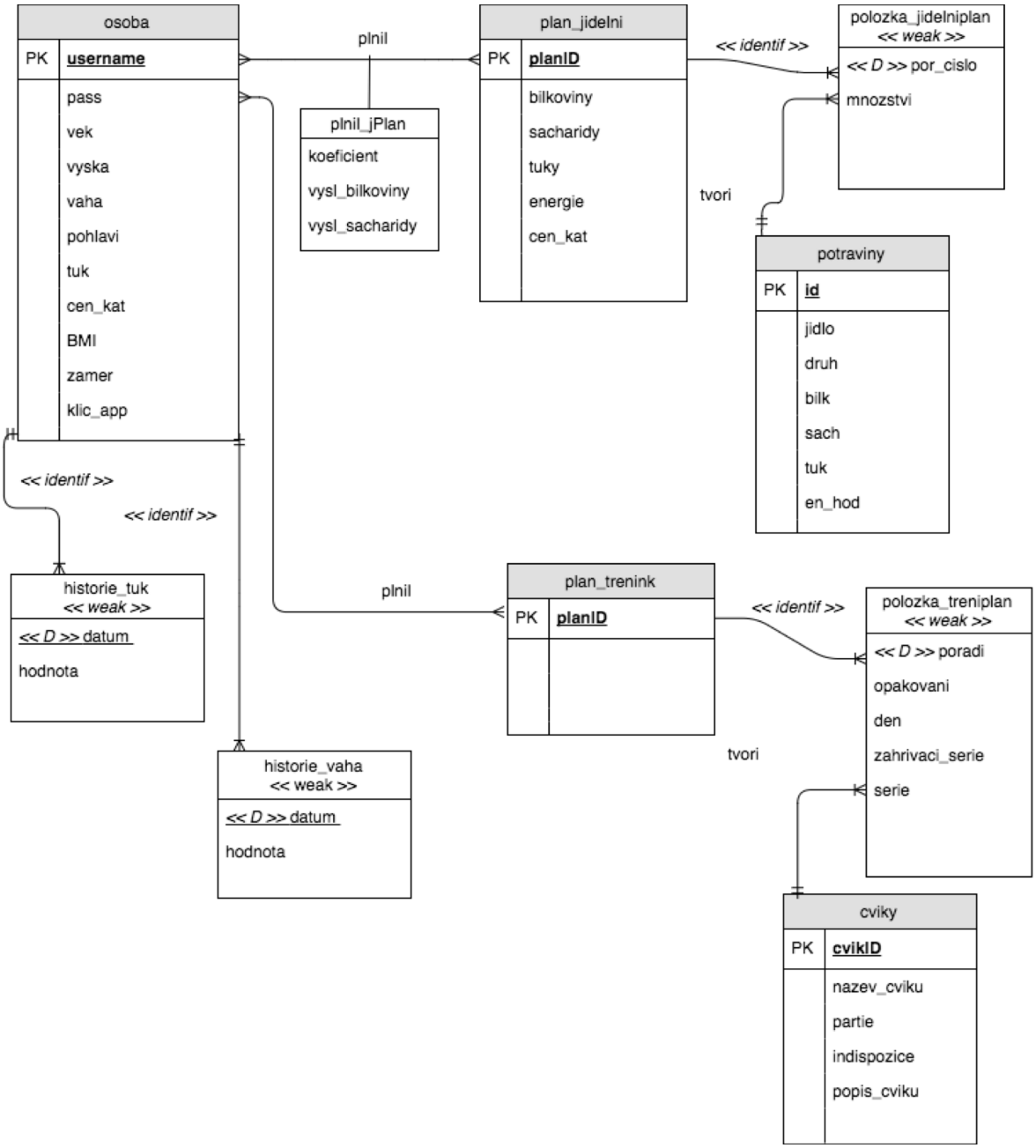
Pro potřeby návrhu databáze jsem využil techniky konceptuálního modelování. Konceptuální model je zcela nezávislý na způsobu fyzického uložení zobrazovaných dat. Z těchto technik jsem si vybral tu pravděpodobně nejpoužívanější, *ER diagram* (z anglického entity-relationship).

Název je složen ze slov

- *entity* – entita, objekt reálného světa,
- *relationship* – vztah.

Je tedy patrné, že v diagramu budou zobrazeny entity. Vztahy budou výsledkem jejich koexistence. Pomocí ER diagramu si lze snadno ujasnit jaká data a v jakém vztahu bude programátor využívat.

Na základě konečné podoby ER diagramu (Obr. 3.5) byl vytvořen konceptuální model databáze.




*Obr. 3.5 ER diagram*

### 3.2.3 Nefunkční požadavky

Důležitým kritériem při návrhu aplikace jsou nefunkční požadavky. Jedná se o skupinu požadavků, které je třeba dodržet pro korektní chod aplikace. Patří sem její výkon, spolehlivost, škálovatelnost, udržitelnost, rozšiřitelnost a modifikovatelnost a další [27].

Ze jmenovaných požadavků jsem mohl při vývoji aplikaci podstatným způsobem ovlivnit pouze její rozšiřitelnost. A to tím způsobem že jsem pevně stanovil počet dietních plánu na **24**. Toto číslo bylo vypočteno kombinatoricky, na základě počtu možností vstupů relevantních pro tento typ plánu, a to následovně: z výšky a váhy uživatele je vypočtena hodnota BMI dle vztahu na Obr. 3.6. Tuto hodnotu dělím do 3 skupin – podváha, normální váha, nadváha (*3 možnosti*). Podle záměru uživatele, který může být buď nabrat nebo zhubnout (*2 možnosti*). Podle cenové kategorie (*2 možnosti*) a podle stavu tělesného tuku (*2 možnosti*), zde je vhodné zmínit, že to, zda je hodnota tuku vyhovující, či nikoliv, ovlivňuje i pohlaví uživatele.

$$BMI = \frac{tělesná\ hmotnost[kg]}{(výška[m])^2}$$

*Obr. 3.6 Výpočet BMI*

U tréninkového plánu je postup analogický. Počet tréninkových plánu uložených v databázi je opět **24**. Nicméně při výběru toho vhodného jsou zohledňovány jiné hodnoty. Jedná se o počet dní, ve kterých chce uživatel vyvíjet fyzickou aktivitu (*3 možnosti*). Věk se dělí na *2 možnosti* (do 15 let věku a nad) a to z toho důvodu, že kostra dítěte je stále ve vývoji a je proto nutné ke stavbě plánu přistupovat jinak, než když se jedná o dospělého jedince. To zda má uživatel problém se zády, který je třeba opět zohlednit při tvorbě plánu, lze vyjádřit *2 hodnotami*. Poslední věc je opět záměr uživatele (*2 možnosti*).

Do kategorie nefunkčních požadavků lze zahrnout i responzivní design. Je totiž velice pravděpodobné (a očekávané), že obsah aplikace, zejména zmíněné plány, budou uživateli často zobrazována přímo během fyzické aktivity ve fitness centru na displejích mobilních zařízení.

# 4 Návrh řešení a implementace

V této kapitole budu popisovat návrh řešení aplikace. Popíši důvody proč jsem si vybral pro implementaci právě tyto technologie.

## 4.1 Technologie webové části

Technologie použité při vývoji odpovídají těm, které byly uvedeny v kapitole 2.2., nicméně je vhodné je detailněji popsat.

### 4.1.1 Volba programovacího jazyka

Pro implementaci webové části aplikace jsem se rozhodoval mezi dvěma skriptovacími jazyky. Uvedu zde jejich charakteristiku a následně své rozhodnutí.

#### Python

Python je programovací jazyk vyšší úrovně, čistě objektový. Obsahuje velké množství datových typů, které je možné rozšířit definováním vlastních. Balík obsahuje v modulech mnoho vestavěných funkcí, například pro práci se síťovými protokoly, databázovými službami a uživatelským rozhraním. Je poměrně dobře přenosný napříč platformami. Může spolupracovat s ostatními jazyky jako skriptovací jazyk.Pro práci s webem je vývojářům k dispozici volně ke stažení množství frameworků.
Rozhodl jsem se některé nejrozšířenější vybrat a popsat je [29].

**Webové frameworky:**

- *Django*
  - zjednodušuje tvorbu webových aplikací, umožňuje vytvářet je s nižšími náklady na množství kódu a rychleji
  - je to vysokoúrovňový rámec, který podporuje rychlý rozvoj a čistý, pragmatický design
  - jedná se o masivní framework s obrovským množstvím doplňků a šablon

- *Flask*
    - tak trochu protiklad Djanga, minimálně rozsahem, jedná se o *mikroframework*, který je pro začátečník velmi vhodný z důvodu strmé křivky učení
    - programátor si musí doinstalovat rozšíření, která potřebuje (např. pokud pracuje s databází, je nutné stáhnout knihovnu, která toto zprostředkuje)

Problémem Flasku je vysoká závislost jednotlivých komponent na sebe. Tím, že nejsou jednotlivé knihovny vydávány pospolu, může snadno dojít k částečné nekompatibilitě mezi jednotlivými součástmi. Je tedy nutné hledat a najít tu verzi knihovny, která bude spolupracovat s ostatními knihovnami.

Django tyto problémy pochopitelně neřeší. Velice často kritizované je ORM (*objektově relační zobrazení* – jedná se o metodu mapování relační databáze na objekty), které degraduje potenciál využití databázového systému [30]. Výhoda tkví ve stejném rozhraní pro vývojáře, kteří se podílí na vývoji konkrétního projektu a také ve snadné kompatibilitě napříč databázemi. Druhý bod je diskutabilní, nestává se často, že by bylo nutné databázi migrovat na jiný systém, než pro který byla navržena.

Tolik tedy ke srovnání rámců.

### PHP

V kapitole 2.3 byl probrán vznik toho jazyku. Od té doby urazil vývoj PHP dlouho cestu, byly přidány mnohé vestavěné funkce a počet uživatelů rostl mílovými kroky. Jako jakýsi milník v historii PHP je považován květen roku 2002 – uvedení PHP 4. Tato verze se od předchozích lišila zejména podporou objektově orientovaného programování, zdokonalením zacházení s prostředky, šifrováním, podporou sezení a několika dalšími vlastnostmi. Vývojáři verze 3.x nepočítali s tak masivním zapojením PHP do rozsáhlých webů a při použití vznikaly problémy. Nová verze tyto nedostatky odstraňovala.

Současná verze, PHP 5, prohlubuje rysy verze předchozí a přidává např. zpracování výjimek typu try/catch a podporu XML (knihovna `libxml2`).

Kód PHP je vykonáván na serveru. Uživatel tedy nemá možnost nahlédnout do zdrojového kódu, protože klientu je odeslána pouze vygenerovaná odpověď zabalená do HTML hlavičky. PHP je nejrozšířenějším skriptovacím jazykem pro web na světe s podílem přes 82%[1]. Další vlastnosti jsou shrnuty v odrážkách:

**Výhody:**

- Vysoká podpora ze strany webhostingů
- Obrovský počet vestavěných funkcí usnadňujících vývoj
- Komunita vývojářů – vzhledem k oblíbenosti jazyka je poměrně snadné řešit problémy spojené s implementací
- Podpora OOP
- Nezávislost na platformě

**Nevýhody:**

- Neexistence ladícího nástroje (lze vyřešit stažením softwaru třetích stran, např. [3])
- Datová náročnost při zpracování (znovunačtení stránky)
- Náročnost na zdroje (zátěž serveru narůstá lineárně s počtem připojených uživatelů)

Zajímavostí pro PHP je *dynamické typování* proměnné. Typ je stanoven na základě obsahu. Tuto vlastnost lze zařadit jak do výčtu výhod tak i nevýhod.

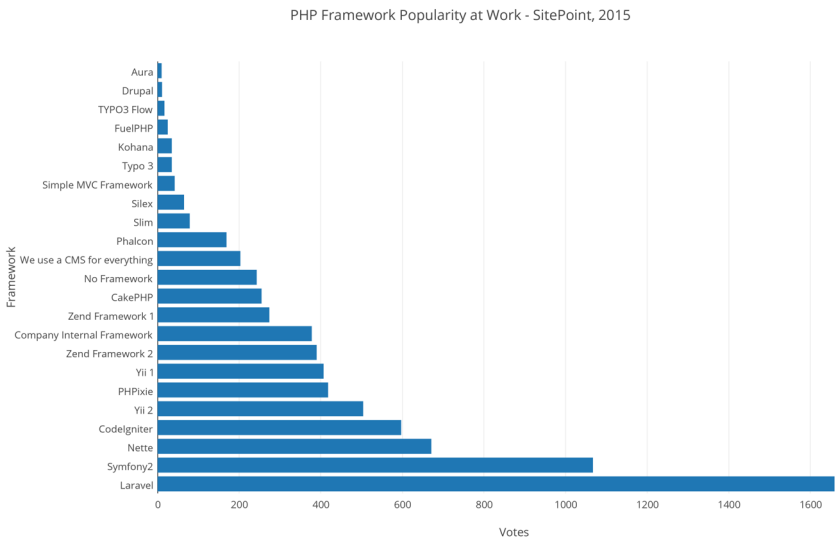
Velmi rozšířené a oblíbené je použití frameworků:

- *Nette*
  - jeden z nejoblíbenějších frameworků v ČR, jedním z důvodů jeho vysoké popularity v tuzemsku je fakt, že se jedná o český projekt; celosvětově jde o 3. nejoblíbenější [12]
  - v testu provedeném serverem *root.cz* obsadil první příčku [14]
  - mezi jeho klíčové vlastnosti patří dobrý výkon a zabezpečení
- *Zend*
  - robustní framework využívaný pro velké projekty
  - mezi firmy využívající jeho služby patří taková jména jako IBM, Microsoft nebo Google [15]
  - obsahuje kryptografické kódovací nástroje, „drag-and-drop“ editor a např. nástroj pro testování PHP jednotek

Nutno podotknout, že existuje značné množství rámců, výše uvedené zástupce jsem vybral na základě všeobecné popularity. Jejich společnou charakteristikou je užití návrhového vzoru MVC, zefektivnění práce, snadnější dodržování dobrých návyků při programování. Na druhé straně, při jejich nasazení dochází ke zpomalení provádění aplikace.

Rozhodovat se mezi Pythonem a PHP jako prostředku pro vývoj webové aplikace je velmi složité. Respektive, finální výběr je spíše otázkou názoru a osobních preferencí, než hledáním pragmatických důvodů proč upřednostnit jeden, či druhý jazyk. Stejný názor jsem našel i na internetových fórech, například [31] [32]. Určitým důvodem pro volbu PHP může být i to, že je rozšířenější, tedy přiklonit se k většině [33]. Já chovám své osobní sympatie k PHP, ačkoliv s Pythonem mám v rámci studijních projektů rovněž zkušenosti. Nicméně, žádné racionální odůvodnění, proč jsme si zvolil PHP jako implementační prostředek, uvést nemohu.

Vzhledem k rozsahu aplikace a nízkému využití potenciálu frameworků jsem se rozhodl pro implementaci v *čistém PHP*.




*Obr. 4.1 Srovnání popularity PHP frameworků[13]*

### 4.1.2 Výběr databázového systému

Rozhodoval jsem se mezi třemi kandidáty. Následuje shrnutí faktů o jednotlivých systémech.

#### SQLite

Jedná se o velmi kompaktní, multiplatformní databázový nástroj napsaný v C. Nabízí mnohé klíčové schopnosti pro správu databází, které poskytují konkurenční produkty, jako jsou MySQL a PostgreSQL. Oproti těmto konkurentům přináší značné úspory v oblasti nákladů, doby potřebné pro jeho zvládnutí, i administračních investic. Mezi hlavní charakteristiky patří následující:

- uložení celé databáze do jednoho souboru, znamená snadnější zálohování či případnou migraci databáze na jiné úložiště,
- veškeré strategie týkající se zabezpečení spočívají v přístupových právech k souboru s databází,
- dostupnost zdarma pro platformy Windows i Unix.

Další pozitivní skutečností je nativní podpora knihovny SQLite v PHP od verze 5 [2].

#### PostgreSQL

*PostgreSQL* je pokročilý objektově-relační databázový systém, je kompatibilní se standardy ANSI/ISO SQL a poskytuje dobrou rozšiřitelnost. Zajímavým řešením, které zajišťuje dodržení ACID vlastností je Multiversion Concurrency Control (MVCC). MVCC funguje tak, že každý uživatel obdrží svou kopii části databáze, se kterou může provádět operace nezávisle na ostatních uživatelích. Hlavní rysy PostgreSQL jsou:

- open-source distribuce,
- podpora objektů,
- rozšiřitelnost pomocí uložených procedur.

Mezi stinné stránky PostgreSQL patří nízká podpora ze strany hostingů, jak je patrné ze srovnávače [15]. Dále je to nižší rychlost při čtení, tedy pomalejší zpracování požadavků, a horší podmínky pro replikaci dat z databáze [16].

#### MySQL

Mezi klíčové vlastnosti MySQL patří zejména výkon, ten je dosažen důslednou optimalizací kódové základny a možností výběru z několika zpracovatelů tabulek (`InnoDB`, `MyISAM`, `BLACKHOLE`...). Programátor při tvorbě databáze volí zpracovatele, který se pro jeho účely hodí nejlépe.

*Cachování dotazů* funguje tak, že MySQL si uchovává dotazy `SELECT` včetně jejich výsledků. Pokud server obdrží dotaz, nejprve je porovnán s obsahem cache, pokud se výsledky shodují, není nutné provádět dotaz nad databází samotnou (což vede ke snížení režie spojené s prohledáváním) a uživatel obdrží výsledek z vyrovnávací paměti.

Aktuální verze MySQL 5.7 obsahuje již od verze 5.1 podporu cizích klíčů, procedur a triggerů. Ve prospěch MySQL mluví i to, že do četných řad uživatelů patří nadnárodní firmy jako Yahoo! Inc., NASA nebo U.S. Census Bureau (Americký úřad pro sčítání lidu). Pro dotazování je využívána mírně pozměněná forma jazyka SQL.

Pro správu MySQL databází lze využít jednoduché grafické uživatelské rozhraní nástroje phpMyAdmin.

V souvislosti s MySQL považuji za vhodné vysvětlit zkratku *LAMP* [17]: je složena z počátečních písmen slov **L**inux, **A**pache, **M**ySQL, **P**HP. Jedná se o běžně užívaný model, kdy spolu spolupracují uvedené technologie a důsledek vysoké míry nasazení MySQL na poli webových aplikací. Tento postup je ověřený mnoha uživateli po celém světě.

Po zvážení těchto tří systémů jsem zvolil *MySQL*. Jedná se o střední cestu hned z několik hledisek. SQLite se díky své přílišné kompaktnosti hodí spíše pro menší projekt s nižším počtem uživatelů. PostgreSQL jsem vyloučil, protože nepotřebuji nutně objektový návrh a také z důvodu pomalejší práce s databází. MySQL se pro mé účely hodí nejlépe.

## 4.2 Výběr platformy pro notifikace na mobilní zařízení

Momentálně na trhu dominují dvě platformy, iOS a Android. Následuje představení, charakteristika a způsob vývoje v obou prostředích.

### 4.2.1 iOS

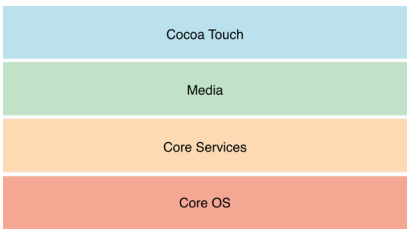
Systém iOS využívají zařízení od společnosti Apple. Jde například o iPhone či iPad. Tato platforma je plně uzavřená, nelze do ní tedy dle potřeby instalovat aplikace třetích stran. Vždy je nutné využít oficiální App Store, který nabízí přes 2 miliony aplikací, v květnu 2017 [18]. Významným kladem operačního systému iOS, potažmo aplikací, pro uživatele je, že návrh, implementace i testování probíhá na jediném zařízení (iPhone nebo iPad) a na tomto zařízení bude i reálně používána. Tento proces má za následek výjimečnou odladěnost a skvělou spolupráci hardwaru a softwaru zařízení.

Každá aplikace dostupná na App Store musela projít schvalovacím procesem, který zkoumá například vizuální podobu designu aplikace. Pokud je vše v pořádku, nic nebrání jejímu zveřejnění.

Vývoj probíhá v Objective-C. Jako vývojové prostředí je využito Xcode integrované na OS X.

#### Architektura iOS

Architektura iOS se dělí na 4 základní vrstvy, které spolu komunikují. Obr. 4.2 ukazuje, jak na sebe navazují.




*Obr. 4.2 Vrstvy iOS*

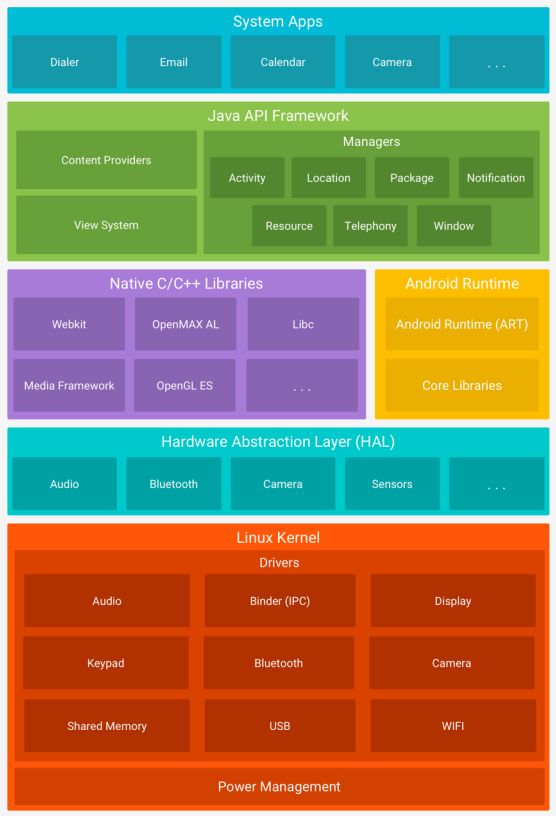
Nejnižší vrstva *Core OS*, zpracovává nízkoúrovňové operace, na těchto operacích je postavena většina dalších technologií. Poskytuje například frameworky pro operace se zařízením bluetooth a akcelerometrem [19]. Vrstva *Core Services* obstarává obsluhu základních služeb jako je iCloud, připojení k jinému zařízení přes bluetooth nebo třeba kopírování dat z iTunes [20]. Vyšší vrstvu *Media*, jak již název napovídá, vývojáři využívají pro práci s multimediálním obsahem [21]. Poslední vrstva *Cocoa Touch* zajišťuje finální vzhled aplikace, tedy její designovou stránku [22].

### 4.2.2 Android

Operační systém Android je vyvíjen společností Google. Je postaven na bázi linuxového jádra a distribuován pod open-source licencí. Jedná se o OS s největší podíl na trhu, tento úspěch je způsoben

právě otevřenou licencí, každý výrobce hardwaru si tedy může do svého zařízení nahrát kopii a zamířit s ním na trh.

Stejně jako iOS má svůj App Store, má Android svůj Google Play. Počet aplikací je ke květnu 2017 mírně vyšší (2 800 000) v porovnání s Applem [23]. Vzhledem k filozofii Googlu, respektive Androidu, není možné, aby byl systém vyvíjen a testován pro každé zařízení zvlášť, tak jako iOS a iPhone. Výsledek tedy může mírně zaostávat v porovnání s „*user experience*“ při používání iPhonu.




*Obr. 4.3 Architektura systému Android*

## Architektura Androidu

Systém Android sestává z pěti vrstev, jejichž názvy budu uvádět v původní anglické verzi, protože neexistuje přesný český překlad. Nejnižší vrstva je samotný *kernel* Linuxu, obsahuje ovladače zajišťující komunikaci mezi hardwarem a softwarem. *Hardware Abstraction layer (HAL)* je složen z několika modulů, tyto mají na starost konkrétní hardwarovou komponentu. Pokud aplikace požaduje přístup např. k fotoaparátu, systém načte modul knihovny zprostředkovávající komunikaci. *Android Runtime(ART)* je poměrně nová součást architektury (od verze 5.0), podstatou je vytvoření instance ART pro každou aplikaci zvlášť ve svém procesu. ART zpracovává bajtkód aplikace do strojového kódu již při její instalaci a tím dochází ke zrychlení při běhu a nižšímu využití akumulátoru. Nevýhodou je déle trvající instalace.
*Nativní knihovny* jsou napsány v C/C++, jedná se o fundamentální funkce systému jako je vykreslení plochy, menu, vrstvení aplikací, atd. Obsahuje také knihovny pro práci s médii a grafikou.
*JAVA API* je propracovaný systém rámců pro vývojáře umožňující např. vytvářet uživatelské rozhraní, tvorbu notifikací a je zodpovědný za správce obsahu napříč aplikacemi. Zajímavé API je např. *Activity Manager*, to zodpovídá za korektní funkci tlačítka *Zpět.* Spravuje tedy „zásobník“ oken jdoucích za sebou v pořadí, jakém je uživatel otevřel.

Poslední vrstva jsou *Systémové aplikace*, což je soubor předinstalovaných aplikací jako Telefon, Zprávy, Kalendář, atp.

Vzhledem ke skutečnosti, že v posledním čtvrtletí roku 2016 tvořily prodeje telefonů s OS Android nebo iOS 99,6 % podíl trhu (jak ukazuje tabulka na Obr. 4.4) , jeví se jako zbytečné uvažovat o dalších platformách jako o potenciálních kandidátech na výběr.

| Operating System | 4Q16 Units | 4Q16 Market Share (%) | 4Q15 Units | 4Q15 Market Share (%) |
|---|---|---|---|---|
| Android | 352,669.9 | 81.7 | 325,394.4 | 80.7 |
| iOS | 77,038.9 | 17.9 | 71,525.9 | 17.7 |
| Windows | 1,092.2 | 0.3 | 4,395.0 | 1.1 |
| BlackBerry | 207.9 | 0.0 | 906.9 | 0.2 |
| Other OS | 530.4 | 0.1 | 887.3 | 0.2 |
| **Total** | **431,539.3** | **100.0** | **403,109.4** | **100.0** |

*Obr. 4.4 Tržní podíl nových mobilních zařízení v závislosti na OS [25]*

Samotný Android ovládá kolem 88% [26] (údaj z listopadu 2016) celkového světového trhu. Hlavní kritérium pro výběr jsem si stanovil počet dosažitelných uživatelů, aplikace bude vyvíjena na *platformě Android*.

### 4.2.3 Výběr platformy pro notifikace

Vzhledem k výběru Androidu, jako prostředí pro vývoj nativní aplikace, bylo nutné vybrat platformu, která bude zprostředkovávat zasílání notifikací na mobilní zařízení. Všechny níže jmenované rozhraní mají společné užití serveru, na kterém jsou *předem* uloženy notifikace, které se odešlou na základě zadaného klíče k aplikaci v případě rozeslání všem uživatelům ve stejném čase (tuto možnost nelze v tomto projektu využít), nebo konkrétní instanci aplikace uložené na určitém zařízení za použití kombinace klíče aplikace a klíče zařízení.

Dalším rysem je využití GCM (Google Cloud Messaging), což je služba poskytovaná společností Google pro všechny zařízení s běžícím systémem Android. Jedná se o zasílání zpráv. Formou zprávy se rozumí i notifikace.

Android SDK obsahuje nativně třídu `NotificationManager`, která je využitelná jen pro možnosti jednoho zařízení (tzn. zaslat notifikaci samo sobě). Tento přístup by byl možný v případě, že by mobilní aplikace obsahovala i funkci zobrazení plánu a následné nastavení notifikace. Kvůli zvolenému přístupu nastavení notifikace z webové části aplikace je tedy nutné vybrat třetí stranu, která bude zaručovat doručení.

#### Parse

První varianta, se kterou jsem při zadání této práce počítal jako s favoritem. Špatnou zprávě je, že majitel společnosti Parse, Facebook, od ledna 2017 oficiálně zrušil provoz serveru. Bylo tedy nutné poohlédnout se po jiné alternativě.

#### Firebase Cloud Messaging

Multiplatformní produkt Googlu, který umožňuje zasílání zpráv a notifikací. Je postavený na GCM. *Klíčové vlastnosti* jsou:

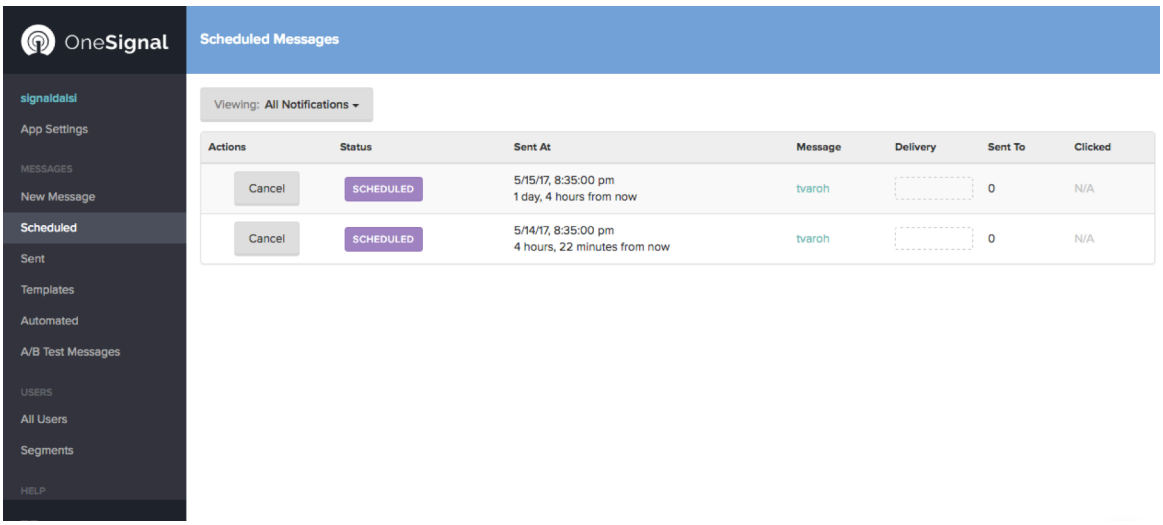
- zasílání běžných textových upozornění a zasílání upozornění, které mohou v mateřské aplikaci zapříčinit nějakou změnu v jejím chodu,
- schopnost distribuovat zprávy jedinému zařízení; skupině zařízení identifikované tzv. tokenem (alfanumerický klíč), každé zařízení v dané skupině je klíčem podepsáno; skupině zařízení identifikované určitou společnost vlastností (např. je možné se přihlásit pro odběr počasí v regionu – společnou vlastností je tedy lokální poloha),
- odeslání zprávy zpět na server – touto cestou je možné získat zprávu, naopak, od zařízení

FCM funguje na následujícím *principu*: Implementace by měla být složena ze dvou samostatných částí.

- *Ověřeného prostředí pro zasílání notifikací* na server Googlu. Nutno podotknout, že je možné zasílat notifikace i přes konzoli na stránkách.
- A pochopitelně samotné *mobilní aplikace*, která vlastní jedinečný identifikační klíč, na základě kterého probíhá doručení zprávy.

Je tedy možné posílat notifikace v podobě požadavku na server. Obsah tohoto požadavku je ve formátu JSON. Až v momentě, kdy jsem měl vše nastavené a implementované, jsem objevil vlastnost nativního GCM bez použití prostředí zmiňovaného v předchozím odstavci, a to je nemožnost naplánování budoucího odeslání notifikace v *předem daný čas*. Tato vlastnost se neslučuje s požadavky aplikace a bylo nutné vyhledat odpovídající řešení v podobě pseudodoručovatele notifikací na GCM server.

Tím se stalo rozhraní **OneSignal**. Umožňuje nejen naplánovat datum a čas odeslání, ale také efektivní správu naplánovaných notifikací pomocí integrovaného webového rozhraní (Obr. 4.5 )




*Obr. 4.5 Ukázka uživatelského rozhraní konzole OneSignal*

# 4.3 Návrh schématu databáze

Z návrhu ER diagramu byl vytvořen návrh schématu databáze. Vlastnosti v ER diagramu aplikace odpovídají vztahům ve schématu databáze.

Konkrétně by tedy aplikace měla obsahovat následující tabulky:

- *osoba* – s údaji o uživatelích, sloupce `vek`, `vyska`, `vaha`, `pohlavi`, `cen_kat` a `tuk` jsou typu `INT`. Údaj `BMI` je `double`. Typu `varchar` jsou údaje `name`, `pass` a `zamer`,
- *plnil_jPlan* – zprostředkovávající údaj o vztahu mezi uživatelem a jím plněným plánem; klíče `username` a `jPlanID` jsou typu `varchar`, respektive `int`, `int` jsou také `vysl_bilkoviny` a `vysl_sacharidy`, `koeficient` je typu `float`,
- *plan_jidelni* – seznam plánů; v této tabulce jsou všechny hodnoty typu `int`,
- *polozka_jidelniplan* – tabulka vyjadřující vztah mezi potravinou a jejím umístěním v konkrétním plánu včetně jejího množství, všechny hodnoty jsou typu `int`

- *potraviny* – banka potravin, ze kterých administrátor čerpá skladbu plánů, sloupec s id je typu `int` a má vlastnost `AUTO_INCREMENT`.

analogicky jsou navrhnuty i tabulky se zaměřením na tréninkový plán

- *plnil_tPlan* s informací o tom, jaký uživatel plní který plán, `username` je `varchar` a `tplan_ID` je `int`,
- *polozka_trenplan* – vyjadřující vztah mezi cvikem a plánem, ve kterým je zahrnut; obsahuje všechny sloupce typu `int`,
- *cviky* – tabulka s uloženými cviky; `cvikID` je typu `int`, má vlastnost `AUTO_INCREMENT`, sloupce `nazev_cviku`, `partie` a `indispozice` jsou typu `varchar` a `popis_cviku` je navržen jak typ `text`.

Tabulky s historií informací o hodnotách tuku a hmotnosti jsou velmi podobné a obsahují:

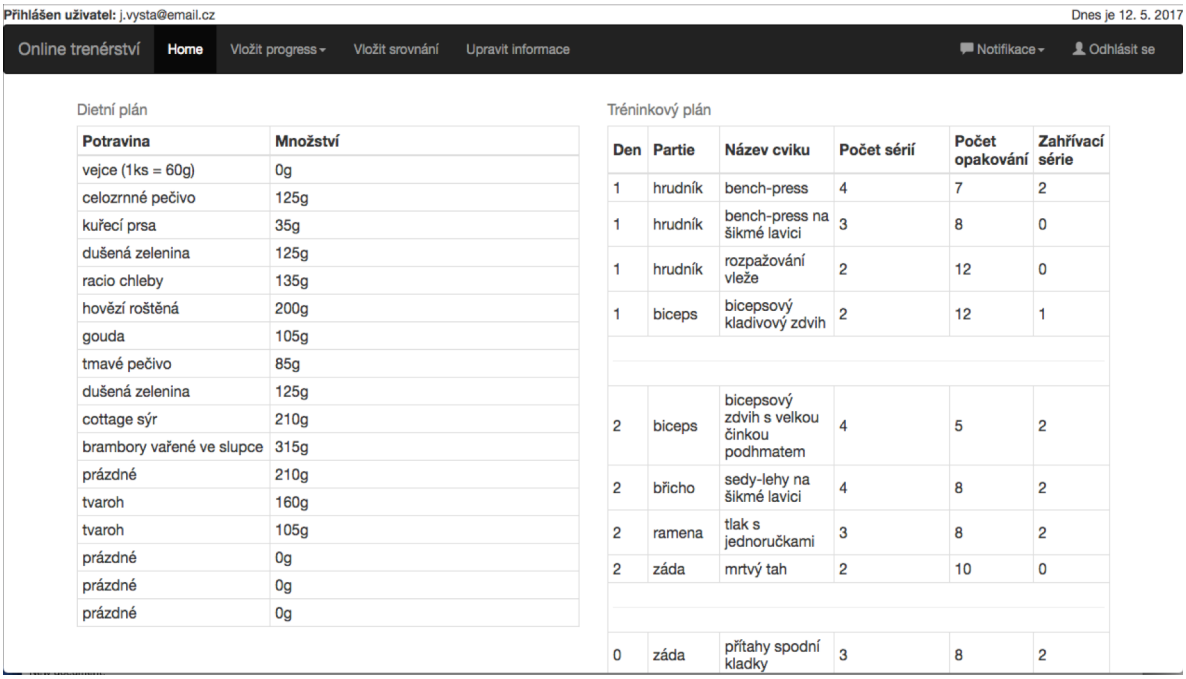
- *historie_tuk* – `username` je typu `varchar`, `datum` je typu `date` a hodnota typu `int`
- *historie_vaha* – `username` je typu `varchar`, `datum` je typu `date` a hodnota typu `int`

## 4.4 Návrh uživatelského rozhraní webové i mobilní části

Primárním cílem při tvorbě uživatelského rozhraní byl responzivní design, čehož bylo dosaženo aplikací knihovny Bootstrap. Volba vzhledu uživatelského rozhraní byla diskutována s kolegou, který se na profesionální úrovni věnuje webovému designu.

Samozřejmostí jsou vlastnosti jako intuitivnost a jednoduchost. Následuje popis výsledného vzhledu, ten byl částečně ovlivněn příspěvky testerů (dále v kapitole Testování).

Základní šablona po přístupu na stránku obsahuje horizontální menu nacházející se na horním okraji, nad tímto menu je aktuální datum a uživatelské jméno (v případě, že je uživatel přihlášen).



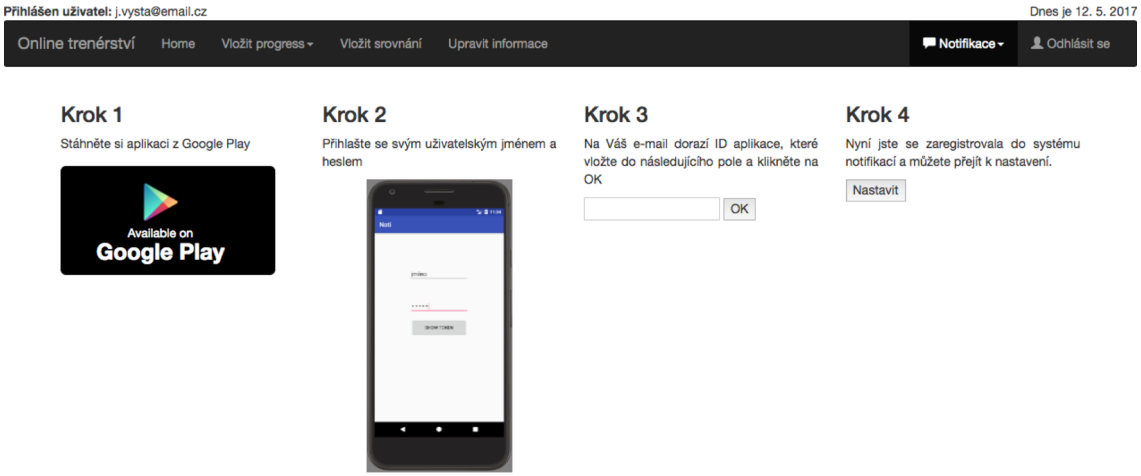

*Obr. 4.6 Uživatelské rozhraní*

Pokud je uživatel přihlášen je úvodní strana rozdělena na 2 vertikální sekce obsahující oba typy plánů přiřazených uživateli – snímek obrazovky se nachází na Obr. 4.6.

Pokud chce uživatel provést vklad aktuálního stavu tuku jsou mu z roletkového menu nabídnuty možnosti volby tuk a hmotnost – obsahy obou obrazovek jsou složeny z vykresleného grafu a formuláře pro vstup nové hodnoty.

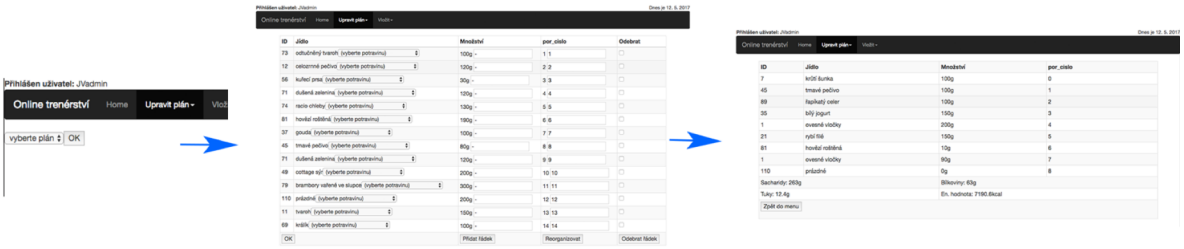
Pro nastavení notifikací je uživateli nabídnuta nápověda ve formě 4 kroků, které jsou orientovány vertikálně vedle sebe (Obr. 4.7)

O něco zajímavější je návrh uživatelského rozhraní pro administrátora, zejména úprava tréninkových plánů (Obr. 4.8).




*Obr. 4.7 Rohraní nápovědy k notifikacím*

Pro tyto účel byly využity 3 obrazovky. Na první je vybrán plán dle čísla. Na další dochází k samotné




*Obr. 4.8 Ukázka nápovědy*

úpravě plánu za použití textových polí a zaškrtávacích políček. Finální produkt úpravy je zobrazen v tabulce v podobě, která je totožná s tím co by viděl běžný uživatel.

Důležitým elementem, který by neměl být při návrhu GUI (grafického uživatelského rozhraní) aplikace opomenut, je vlastnost vizuální zpětné vazby. Tedy pokud uživatel klikne na nějaký prvek, měla by mu aplikace dát vědět, že tento jeho akt byl úspěšně zaznamenán a je zpracováván. Pokud by aplikace takto nebyla navržena, může se stát, že uživatel např. v e-shopu několikrát klikne na tlačítko a vloží si do košíku 5 položek namísto jedné. Je to výhodné i v případě, že když uživatel ví, že má zpětnou vazbu očekávat a žádná se neprojeví, je pravděpodobné, že má problém s prohlížečem.

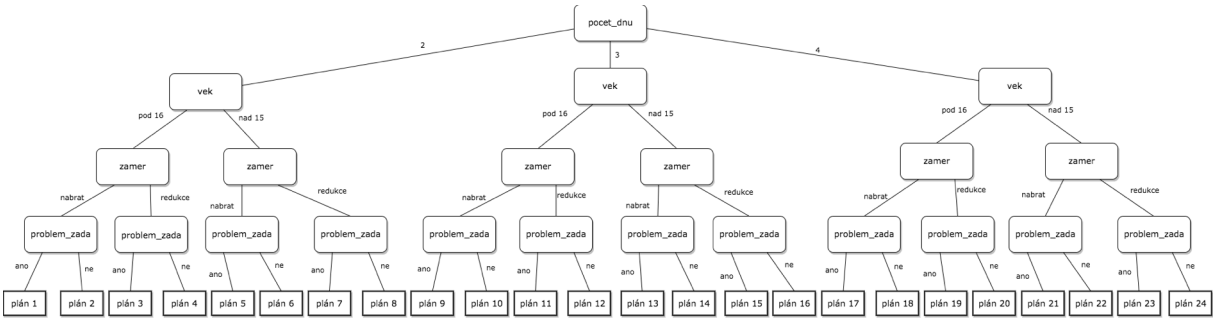
## 4.5 Popis implementace a realizace případů užití

V této kapitole popíši metody aplikované při implementace. Ještě než přistoupím k vysvětlení implementace několika vybraných případů užití, je nutné představit koncepci aplikace.

Tato práce je rozdělena na webovou část spolupracující s databází a mobilní část tvořenou nativní Android aplikací.

### 4.5.1 Webová část

Z požadavků na aplikaci je patrné, že fundamentální funkcí aplikace je přiřazení plánu uživateli. Prvním krokem tedy je, na základě vstupních informací, rozhodnout o výběru plánu. Toho bylo dosaženo




*Obr. 4.8 Rozhodovací strom aplikace*

převedením všech hodnot z formuláře do číselné podoby a jejich následným postupným vyhodnocováním. Tato část souvisí s požadavky uvedenými v kapitole 3.2.3, důležitým poznatkem, že tento proces vyhodnocování je velice podobný *stromové hierarchii*.

Na Obr. 4.9 je zobrazena stromová hierarchie při rozhodování o přidělení tréninkového plánu. Je možné vidět, že nejdřív se řeší otázka počtu dnů, pokud uživatel zvolil např. možnost 2, je stejný postup iterován s hodnotou věku v levém podstromu. Tento postup je tedy čtyřikrát opakován, než se program dostane na listovou úroveň, ve které se nachází číslo výsledného plánu. Na úrovni zdrojového kódu se jedná o zanořené větvení. Analogický postup, s jinými parametry, je proveden při volbě dietního plánu. *Struktura zdrojových kódu* více či méně odpovídá jednotlivým funkcionalitám aplikace. To znamená, že pro jeden úkon uživatele (např. vkládání osobních informací) existuje jeden `*.php` soubor, se kterým uživatel pracuje. Pro přístup k databázi využívá aplikace objektové rozhraní *PDO* (PHP Data Objects) [35].

Popis implementace webové části je vhodné rozdělit na tři části dle modelu případů užití – neregistrovaný uživatel, registrovaný uživatel a administrátor.

*Registrace uživatele* probíhá tak, že po korektním vyplnění údajů ve formuláři je v tabulce `osoba` vytvořena nová položka. Tato položka obsahuje pouze uživatelské jméno s prázdnými sloupci pro ostatní informace. Po zaregistrování se uživatel může přihlásit do systému. Po úspěšném přihlášení je dialogovým oknem upozorněn, že o něm nejsou známy žádné osobní údaje, nutné pro tvorbu plánů, a je přesměrován na stránku s formulářem pro jejich vložení. Odesláním formuláře je vyžádán plán, jehož selekce probíhá výše zmíněným způsobem.

Pro vložení pokroku v tréninkovém plánu je nutné vybrat si z roletkového menu, možnost tuk/váha. Z pohledu souborové struktury se jedná se o dva rozdílné moduly, `vlozit_progress_tuk.php` a `vlozit_progress_vaha.php`. Při implementaci jsem využil javascriptovou knihovnu pro vykreslování grafů CanvasJS [34]. Použitím této knihovny je do stránky vložen kontejner, se jedná o HTML element `<div>`, do kterého je uložen nově vzniklý objekt typu `CanvasJS.Chart`. Při vzniku jsou tomuto objektu přiřazeny parametry (ve formátu JSON) grafu, který má obsahovat. Tyto parametry jsou složeny z částí datové, obsahující informace z databáze –

hodnoty váhy, případně tuku, vztahující se ke konkrétnímu časovému údaji. Tyto hodnoty tvoří osu $x$ a $y$. Nakonec je voláním vestavěné funkce `render()` celý graf vykreslen. Pod ním se nachází formulář pro tvorbu nových záznamů. Zde bylo nutné ošetřit správný vstup, konkrétně formát data. Ten je kontrolován regulárním výrazem – vestavěnou PHP funkcí `preg_match()`, vracející hodnotu `true`, pokud dojde ke shodě nad řetězcem, který je zavolán jako parametr funkce. V případě nevalidního vstupu je uživatel upozorněn na chybu dialogovým oknem JavaScriptu. Pokud vše proběhne v pořádku, nové údaje jsou uloženy do databáze a je zobrazen aktualizovaný graf.

Obrazovka sloužící pro *vložení srovnání* s dietním plánem je složena z výpisu plánu z databáze přiřazenému uživateli (tato informace je čerpána z tabulky `plnil_jPlan`). Tato tabulka má možnost zaškrtnutí „Splněno“ značící, že uživatel zkonzumoval potravinu. Postup výpočtu je poněkud komplikovanější, než by se mohlo zdát, proto ho zde vysvětlím. Obsah všech plánů je de facto uložen v tabulce `polozka_jidelniplan`. V této tabulce se nachází číslo plánu, pořadí potraviny v tomto plánu, její název a množství. Důležitým poznatkem je, že obsah všech plánů je dimenzován na osobu vážící 90kg. V případě, že aktuální uživatel má ve sloupci `vaha` hodnotu např. 72kg, je nutné všechny hodnoty množství přepočítat dle vztahu na Obr. 4.10. Tento koeficient (72/90 = 0.8) je uložen do `plnil_jPlan` již při prvotním zadávání osobních informací a při každém výpisu dietního plánu je aplikován na sloupec množství, kterým je vynásoben. Vynásobením vznikají desetinná čísla, mnohdy s periodickým desetinným rozvojem, jejichž přesnost je v kontextu zcela irelevantní, přistoupil jsem tedy k zaokrouhlení této hodnoty a nastavení nejmenší možné odchylky na 5g.

$$koeficient = \frac{váha\ uživatele[kg]}{90}$$

*Obr. 4.9 Výpočet koeficientu*

Současně je tímto koeficientem ovlivněna i hodnota celkového počtu bílkovin a sacharidů v plánu. Pokud tedy uživatel zaškrtne, že splnil, aplikace vybere z tabulky `polozka_jidelniplan` hodnoty množství, vynásobí je koeficientem a sečte je. Celková suma těchto hodnot je podělena hodnotou bílkovin/sacharidů, uvedených v tabulce `plnil_jPlan`, a je uživateli vypsána procentuálně. Pro větší uživatelský komfort byla pomocí JavaScriptu implementována možnost „Zaškrtnout vše“. Problematika realizace notifikací bude rozebrána v dalších kapitolách.

Uživatel s právy administrátora má, pochopitelně, více možností práce s aplikací. Jedná se zejména o úpravu plánů obou typů. Protože se oba postupy mírně liší, vysvětlím nejdříve implementaci úprav dietního plánu. Postup je takový, že administrátor z roletkového menu vybere číslo plánu, který míní upravit. Z databáze, konkrétně z tabulky `polozka_jidelniplan`, jsou v tabulce vypsány její hodnoty společně s formulářovými prvky pro editaci. Pro výběr potraviny je k dispozici roletkové menu, do kterého jsou informace čerpány z tabulky potravin. Tato menu jsou u každé položky. Při volbě názvu je nutné brát v úvahu, že stejná potravina se v plánu může vyskytovat 0 až n-krát. Zvolil jsem tedy způsob vyobrazený kódem na Obr. 4.11. Při následném zpracování dat z formuláře je použita funkce `explode($delimiter, $string)`, která rozdělí, na základě hodnoty oddělovače, předaný řetězec do proměnné typu pole. Hodnota získaná tímto způsobem je využita pro přepis (`UPDATE`) položky tabulky nalezené dle primárního klíče `id_planu` + `por_cislo`. Úprava sloupce množství probíhá obdobným způsobem, jen z důvodu velkého množství potenciálních hodnot není zvolena možnost roletkového menu. Nový údaj uživatel vepíše do textového pole.

Tlačítko *Přidat řádek* vytvoří v tabulce novou položku s hodnotami aktuální `cislo_planu`, a protože se nová položka vkládá implicitně na konec, má sloupec `por_cislo` hodnotu o jedničku vyšší než dosavadní poslední prvek. Pak už s tímto novým řádkem lze pracovat jako s ostatními.

```
1.  <select name =".$rows['id']."_".$rows['por_cislo']."> 
```

*Obr. 4.10 Element select pro výběr potraviny*

Dále má uživatel možnost *Odebrat řádek*. Postup při implementaci je takový, že nejdříve je daný řádek smazán a následně je pomocí `RowCount()` zjištěn počet řádků. Celý plán je poté znovu očíslován hodnotami 1 až `RowCount()`. Tímto postupem je zajištěna integrita a návaznost jednotlivých položek na sebe.
Tlačítko *Reorganizovat* znamená změnu pořadí potravin. Pokud uživatel v tomto poli zadá např. hodnotu 2, je tento řádek přemístěn na druhé místo v plánu zatímco původní druhý prvek je odsunut na místo třetí. Tento postup dává větší smysl, než kdyby se jednalo o prosté prohození položek. Implementačně se jedná o 3 příkazy `UPDATE`. Pokud se jedná o případ, kdy je řádek řazen v tabulce směrem dolů (*puvodni_cislo* < *nove_cislo*), je nejdříve nastaveno pořadové číslo (sloupec `por_cislo`) na 0; poté je snížena o 1 hodnota stejného sloupce položek nacházejících se mezi hodnotami *puvodni_cislo* a *nove_cislo*. Posledním krokem je změna pořadí měněné položky z 0 na *nove_cislo*.

Vkládání *nového cviku* spočívá ve vyplnění formuláře, jehož hodnoty jsou uloženy do databázové tabulky `cviky`. Ještě před příkazem `INSERT` probíhá ověření, zda jsou zadány korektně všechny položky, např. nesmí být vyplněna hodnota pro název nové partie zároveň se zvolenou možností v roletkovém menu. Na případné problémy je uživatel opět upozorněn dialogovým oknem JavaScriptu. V případě vkládání informací o potravinách je postup obdobný jako u cviků.

Možnost *prohlížet* a následný výběr potravina/cvik obsahuje výpis položky z databáze, včetně uložených informací. Volba *smazat potravinu* dostupná na obrazovce s náhledem, poskytuje kontrolu, která hlídá zda se potravinu, kterou uživatel zamýšlí smazat, nevyskytuje v jiných plánech.
Pokud ano, upozorní na tuto skutečnost uživatele, vypíše seznam těchto plánů a zeptá se, zda si přeje smazat ji i přes tuto skutečnost. Obsah zmíněných položek je následně smazán a nahrazen prázdnou hodnotou.

## Grafické uživatelské rozhraní webové části

Jak již bylo zmíněno, grafické uživatelské rozhraní bylo implementováno pomocí prvků HTML a jejich formátováním v CSS. Odpovídá návrhu z kapitoly 4.4. Primárním požadavkem byla jednoduchost a grafická čistota. Základním prvek UI (*user interface* – uživatelské rozhraní) je horizontální menu v záhlaví všech stránek. Tento prvek je tvořen HTML5 značkou `<nav>`, která slouží pro tvorbu navigačních odkazů. Její původní vzhled byl od základů změněn pomocí tříd `navbar` a `navbar-inverse` rámce Boostsrap. Významným prvkem menu jsou roletky použité pro výběr možností tuk/váha a u administrátorského účtu pro úpravy plánů, katalog cviků a potravin a jejich vkládání. Pro tyto účely byla využita bootstrapová třída `dropdown`.

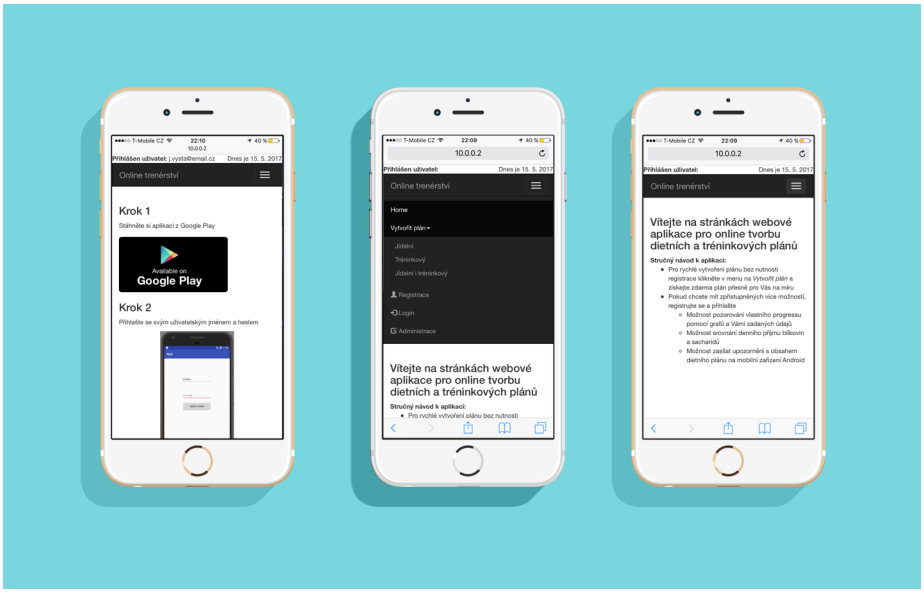
Vzhledu mnohokrát zobrazované tabulky bylo dosaženo použitím následujících tříd: „`table table-hover table-condensed table-striped table-bordered`“. `table-hover` je důsledkem toho, že při najetí kurzoru nad buňku tabulky je lehce zvýrazněn celý řádek. Díky `table-striped` jsou sudé a liché řádky barevně odlišené.

Výhodou Boostrapu je, že nativně podporuje již dříve několikrát zmíněny responzivní design.

```
1.  <div class="container">
2.  <!--Responzivní obsah-->
3.  </div>
```

*Obr. 5.1 Tvorba responzivního obsahu*

V případě aplikace byl dosažen uzavřením celého viditelného obsahu stránky mezi počáteční a koncovou značku elementu `<div>` (Obr. 5.1). Výsledný vzhled na mobilním telefonu o uhlopříčce 4,7“ je na Obr. 4.12. Snímky obrazovky notebooku o úhlopříčce 13“ jsou vidět na Obr 4.6, Obr. 4.7 a Obr. 4.8.




*Obr. 4.11 Ukázka responzivního designu aplikace*

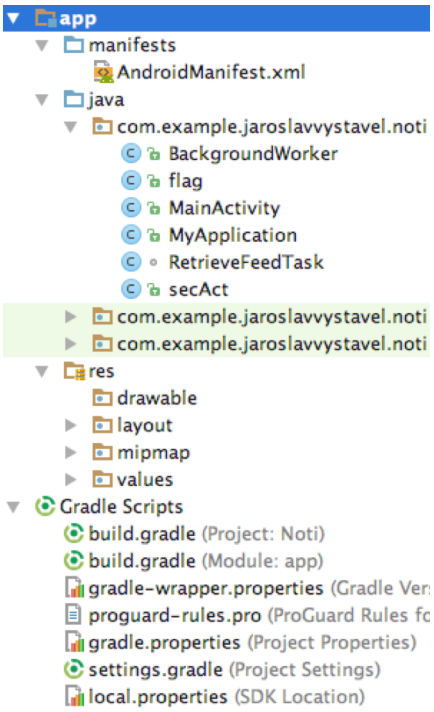
### 4.5.2 Mobilní část – notifikace

V kapitole 4.2 jsem si pro vývoj mobilní části aplikace vybral platformu Android. Tato kapitola obsahuje popis implementace a informace o použitých technikách a o způsobu kooperace vyvíjené aplikace a služby OneSignal. Vzhledem k účelům aplikace nebylo nutné složitě navrhovat uživatelské rozhraní, proto se zde tato část nevyskytuje.

Vývoj probíhal v přirozeném vývojovém prostředí od společnosti Google – Android Studiu. Jako implementační nástroj byl použit jazyk Java, který je standardem pro tento OS. Při zadání práce nebylo specifikováno pro kterou verzi systému má být aplikace vyvíjena, tak jsem na základě vlastního rozhodnutí zvolil API 17 (*Application Programming Interface* – rozhraní pro programování aplikací), které odpovídá Androidu 4.2. Toto API vyšlo koncem roku 2012 [36]. Důvodem mé volby je fakt, že při vývoji na starších verzích je zajištěna zpětná kompatibilita s verzemi novějšími. Naopak benefitem výběru aktuálnějšího vydání je dostupnost novějších funkcí a služeb, které dříve neexistovaly. Nicméně, tato mobilní aplikace nemá ambice užívat nejnovější technologie, proto si vystačí i se starším rozhraním.

Smyslem aplikace je získat identifikační alfanumerický kód, který bude sloužit jako adresu pro zařízení, na kterém je aplikace nainstalována. Uživatel tedy vyplní přihlašovací jméno a heslo, tyto údaje jsou totožné s těmi, které používá ve webové verzi aplikace, proběhne ověření vůči databázi a na e-mail, který slouží zároveň jako login, je odeslán zmíněný kód. Tento kód uživatel vloží do webové aplikace a tím je vložen do databáze.

### Souborová struktura aplikace




*Obr. 4.12 Adresářová struktura projektu*

Při vytvoření nového projektu je pro vývojáře předpřipravena souborová struktura do jejichž částí více či méně zasahuje (Obr. 4.13). Některé podstatné adresáře a soubory zde zmíním.

Soubor `AndroidManifest.xml`, nacházející se v kořenovém adresáři každého projektu, obsahuje důležité informace pro zařízení, na kterém bude aplikace spuštěna. Jedná se například o ikonu, zobrazovaný název a také jsou zde formulována práva aplikace, jako je přístup k internetu, propojení jednotlivých tzv. *aktivit* (jedná se o základní kámen aplikace, slouží pro vykreslení elementů na obrazovku a popis způsobů jejich interakce) a také definovanou aktivitu, který má být spuštěna jako první.

Adresář `layout` obsahuje vizuální rozvržení jednotlivých aktivit ve formátu XML, obsahuje tedy počet souborů odpovídající počtu aktivit. Určuje finální podobu grafického uživatelského rozhraní. V adresáři `values`, jsou uloženy údaje v podobě `klíč:hodnota`, tyto hodnoty jsou odkazovány dalším souborům, které je používají.

`build.gradle` shromažďuje informace pro kompilátor, jsou zde, mimo jiné, importy různých knihoven a informace o využívaných službách.

Následuje výpis hlavních modulů (tříd), které jsem implementoval v projektu aplikace:

- `MainActivity` – jedná se o hlavní třídu aplikace, která je spuštěn po jejím otevření. Zde se nachází základní funkcionalita aplikace. Využívá elementy definované v `activity_main.xml`. Jedná se o `<EditText`, potomka veřejné třídy `EditText`, textová pole pro uživatelský vstup a `<Button`, tlačítko. Na toto tlačítko je nastaven posluchač událostí (*event. listener*), přesněji se jedná o `OnClickListener(new View.OnClickListener() {...}`. Tento slouží jako spouštěč událostí.
- `secAct` – je druhá a poslední aktivita. Uživateli je zobrazena až po úspěšném přihlášení a obsahuje pouze element `textView` s uvítací zprávou, informující uživatele, a tlačítko pro odhlášení ze systému. Její vizuální obsah se nachází v `activity_sec.xml`.
- `RetrieveFeedTask` – skládá se z generické asynchronní třídy `AsyncTask`. Je užitečná zejména v tom, že programátor nemusí ručně vytvářet vlákna a obstarávat je. Instance této třídy bývá většinou spouštěna na místech, kde není nutné čekat na její odpověď v tom smyslu, že ji programátor nepotřebuje ihned. Výhodou je, že uživatel je ušetřen zdánlivého zaseknutí provádění aplikace. V této aplikaci ji využívám pro síťovou komunikaci. Ve funkci `doInBackground()` probíhá tvorba požadavku, za použití knihovny `OkHttp` je vytvořen nový požadavek, jako instance třídy `OkHttpClient`, a do jeho těla (`body`) je vložen obsah, který je odeslán na server pro rozeslání e-mailu s id pro notifikace.

### Aktivity

Prvním krokem je uložení přihlašovacích údajů do proměnných odeslání na server s databází, kde proběhne jejich ověření. Samotný požadavek je vytvořen pomocí Java knihovny `Volley`, která slouží pro práci se sítěmi. Podoba požadavku je ve formátu JSON, který je nezávislý na platformě a jehož data jsou organizována v podobě objektů.
Pokud je vrácen *úspěch*, jako odpověď serveru, následuje volání metody `idsAvailable` objektu OneSignal (Obr. 4.14).

```
1.  public void zjisti(){
2.   OneSignal.idsAvailable(new OneSignal.IdsAvailableHandler() {
3.
4.       @Override
5.       public void idsAvailable(String userId, String registrationId) {
6.           Log.d("debug", "User:" + userId);
7.           new RetrieveFeedTask().execute(userId,email.getText().toString());
8.           Toast.makeText(MainActivity.this, userId, Toast.LENGTH_SHORT).show();
9.           if (registrationId != null)
10.              Log.d("debug", "registrationId:" + registrationId);
11.      }
```

*Obr. 4.13 Volání metody idsAvailable()*

`player_Id`, jak OneSignal nazývá identifikační 32místný alfanumerický kód, slouží jako cílová adresa doručení notifikace a je jedinečný pro každou aplikace spuštěnou na stejném zařízení. Tuto hodnotu uživatel, vložením do pole ve *webové částí,* uloží do databáze.

Po tomto prvotním nastavení, je uživatel schopný spravovat agendu notifikací. Podoba výsledného požadavku je opět ve formátu JSON. Ten obsahuje mj. položky pro nastavení času, textového obsahu a právě zmiňovaného `player_id`. HTTP požadavek je sestaven pomocí PHP knihovny *Client URL Library* – `curl`.

# 5 Testování

Smyslem testování je odhalit chyby a nedostatky vzniklé při implementace. K testování jsem zvolil dva přístupy. Prvním bylo verifikační testování, které jsem prováděl já již během vývoje aplikace pro odladění zjevných chyb, dalším je testování na reálných uživatelích.

Způsobů uživatelského testování je několik, např. nechat uživatele o samotě s aplikací, tak aby byla simulována reálná práce s aplikací, stejně jako v případě reálného nasazení. Dalším způsobem je pozorovat testera při činnosti interakce s aplikací a sám si tvořit poznámky, na základě jeho reakcí na určité situace. V obou případech je vhodné nechat si od něj vyplnit formulář týkající se spokojenosti s aplikací.

### Verifikační testování

Průběh verifikačního testování se dá rozdělit na tři etapy, webová aplikace, uživatelské rozhraní, agenda notifikací. Z pojmenování těchto etap je evidentní, co bylo jejich předmětem.

Ve všech těchto případech bylo nutné se vžít do pocitů člověka, který aplikaci v životě neviděl. Stejně tak bylo nutné postavit aplikaci tak, aby i uživatel pohybující se ve fitness neměl problém s jejím užíváním (tzn. aby uživatel nemusel být zkušený programátor).

Výsledným efektem těchto testů bylo např. to, že jsem se rozhodl změnit polohu menu z vertikálního na horizontální. Když se menu nacházelo v původní poloze, docházelo k esteticky nepříjemnému prázdnému místu v oblasti pravé strany obrazovky. Bootstrapové tabulky sice umožňují přizpůsobit velikost obrazovce, ale výsledný dojem na 21“ displeji monitoru nebyl zrovna pozitivní. Tento neduh byl odstraněn výše zmíněným řešením. Ve výsledku je tedy obsah zobrazován do středu obrazovky. Opticky to působí přirozeněji.

Po sérii těchto testů, kdy docházelo zejména k úpravám uživatelského rozhraní, jsem si domluvil schůzky s testery a přestoupil k další fázi testování.

### Uživatelské testování

Účelem uživatelského testování je zisk zpětné vazby od reálných osob, které dosud nepřišli s aplikací do styku. Ideální vzorek těchto osob je cílová skupinu uživatelů, kteří mají předpoklad, že se do konfrontace s aplikací opravdu dostanou. Skladba testerů by měla být dostatečně rozmanitá, ale přitom by mělo zůstat zachováno pravidlo zmíněné v minulé větě.

Postup při testování byl takový, že jsem po testerovi vyžadoval, aby se vžil do situace, která by ho vedla k použití aplikace (např. změna životního stylu, touha udělat něco se svou postavou). Při samotné činnosti testování jsem se snažil co nejméně zasahovat do úkonů testera. Stalo se tak až v opravdu kritický případech, kdy uživatel doslova nevěděl, co má dělat. Jako cíl jsem stanovil, aby si uživatel postupně prošel všechny případy užití. Jeho počínání jsem si pečlivě zapisoval a přemýšlel o možných zlepšeních. Závěrem byly otázky typu „Co lze zlepšit? “, „Co vás při použití mátlo?“, „Co se vám líbilo?“, „Co se vám nelíbilo?“, „Co byste z aplikace odstranili?“.

Výsledky jsem shrnul do následující tabulky (Obr. 5.14). V prvním sloupci je charakteristika osoby. Jako relevantní aspekty jsem vyhodnotil:
*věk* (mladší generace má k moderním technologiím obecně o něco blíže), *aktivita ve fitness* (souvisí zejména s orientací v pojmech použitých v UI) a *znalost IT* ( která by měla pomoci při práci s aplikací). Ve sloupcích silné/slabé stránky se vyskytují reakce na zmíněné otázky. Po testerech jsem vyžadoval aby známkou 1-10, přičemž 10 znamená nejlepší, ohodnotili výsledný dojem z používání aplikace.

## Závěr uživatelského testování

Do sloupce *poznámka* jsem si uváděl své osobní postřehy, které nebyly konfrontovány s testerem. Pomohly mi však při následných úpravách aplikace. Několik testerů mělo problém při orientaci v aplikaci. Řešením tohoto problému bylo přidání textové nápovědy na první stránku aplikace.

Dalším poměrně frekventovaným problémem byly chyby a nejasnosti při vyplňování formuláře na stránce pro zaznamenání pokroku ve fitness plánu (tuk/váha), jedná se totiž o anglický formát zápisu a to uživatele mátlo. Tento problém jsem eliminoval malou textovou nápovědou v řádku se vstupem pro datum.

Problémy, jako obrázky na úvodní stránce a malý počet barev v menu, jsem zanedbal, protože se jedná individuální dojem testera, který je spíše otázkou vkusu.

| Profil testera | Aplikace | | Celková spokojenost | Poznámka |
|---|---|---|---|---|
| | Silné stránky | Slabé stránky | | |
| muž, 25 let, aktivní ve fitness, bez znalosti IT | přehlednost plánů; systém notifikací | formát datumu, více barev, obrázek do pozadí | **9/10** | Tester měl nejvíce problémů s formátem datumu; dokonce došlo k záměně nuly za písmeno "O" |
| žena, 28 let, aktivní ve fitness, bez znalosti IT | přehledný,jednoduchý vzhled; celkové množství služeb | způsob nastavení notifikace | **9/10** | Tester byl poněkud zmaten ze způsobu nastavení notifikací |
| muž, 23 let, aktivní ve fitness, znalost IT | celkový dojem z aplikace, její smysl a nápad | potřeba manuálně kopírovat ID z e-mailu do kolonky v aplikaci (detail) | **10/10** | Tester bez problémů prošel všechny zamýšlené případy užití |
| žena, 45 let, neaktivní ve fitness, znalost IT na uživatelské úrovni | líbivé služby, které aplikace nabízí | "nízký počet barev uživatelského rozhraní" | **10/10** | Testování proběhlo bez větších komplikací |
| muž, 48 let, aktivní ve fitness, bez znalost IT | nápad; služby aplikace | orientace v aplikaci; neznalost pojmu notifikace | **8/10** | Nejvetším problémem byla orientace v aplikaci |

*Obr. 5.14 Tabulka s výsledky testování*

# 6 Závěr

V rámci této bakalářské práce byla navržena a implementována webová aplikace umožňující individuální tvorbu fitness plánů. Kromě této základní funkcionality aplikace dále umožňuje vizuálně znázorňovat pokrok v tomto plánu v rámci dlouhodobého období, možnost srovnat úroveň splnění dietního plánu pomocí webového formuláře a také možnost nastavení notifikací pro mobilní zařízení s operačním systémem Android.

Webová aplikace je na klientské straně implementována za použití HTML, CSS, JavaScriptu a javascriptového rámce CanvasJS. Pro designové účely a zajištění responzivity byl využit rámec Bootstrap. Na straně serveru se jedná o technologie PHP a MySQL.

Testování proběhlo na několika uživatelích. Relevantnost testování byla podpořena vhodným výběrem testerů, se kterými byly vedeny osobní schůzky. Na základě těchto schůzek byla aplikace přetvořena do finální podoby.

Potenciál této aplikace je, troufám si tvrdit, poměrně značný. Po patřičném zásahu do obsahu databáze a stavby jednotlivých plánů, ze strany školeného trenéra, nic nebrání plnému nasazení aplikace do ostrého provozu. Umím si představit, že aplikace bude působit na webových stránkách fitness centra, jako levná alternativa placeným osobním trenérům. Je pravda, že nic nemůže nahradit trenérův osobní přístup a jeho úsudek, nicméně ambicí této aplikace není brát práci trenérům, ale je mířena spíše na začátečníky, kteří by služeb trenéra stejně pravděpodobně nevyužili.

# Literatura

[1] *W3Techs* [online]. [cit. 2017-05-17]. Dostupné z: https://w3techs.com/technologies/overview/programming_language/all

[2] GILMORE, W. J. *Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály*. Brno: Zoner Press, 2005. Encyklopedie webdesignera. ISBN 80-868-1520-X.

[3] *XDEBUG EXTENSION FOR PHP* [online]. [cit. 2017-05-17]. Dostupné z: https://xdebug.org/

[4] ZAKAS, Nicholas C. *Professional JavaScript for web developers*. 3rd ed. Indianapolis: Wiley, c2012. ISBN 11-180-2669-1.

[5] CASTRO, Elizabeth a Bruce HYSLOP. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

[6] PÍSEK, Slavoj a Bruce HYSLOP. *HTML: začínáme programovat*. 4., aktualiz. vyd. Praha: Grada, 2014. Průvodce (Grada). ISBN 978-80-247-5059-0.

[7] *Bootstrap-file-structure.jpg* [online]. [cit. 2017-05-17]. Dostupné z: https://etouchdesignteam.files.wordpress.com/2013/01/bootstrap-file-structure.jpg

[8] SAYED SALEH, Mukhtar a Mayed SAYED SALEH. *Mobile Applications Development: Using Phonegap*. ISBN [ISBN neznámé].

[9] *Bootstrap History* [online]. [cit. 2017-05-17]. Dostupné z: https://v4-alpha.getbootstrap.com/about/history/

[10] KONOPKA, Peter. *Sportovní výživa*. České Budějovice: Kopp, 2004. Průvodce sportem. ISBN 80-723-2228-1.

[11] COCKBURN, Alistair. *Use Cases: jak efektivně modelovat aplikace*. Brno: CP Books, 2005. ISBN 80-251-0721-3.

[12] *The Best PHP Framework for 2015: SitePoint Survey Results* [online]. [cit. 2017-05-17]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

[13] *PHP Framework popularity* [online]. [cit. 2017-05-17]. Dostupné z: https://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2015/03/1427547421php_framework_popularity_at_work_-_sitepoint2c_2015.png

[14] *Velký test PHP frameworků: Zend, Nette, PHP a RoR* [online]. [cit. 2017-05-17]. Dostupné z: https://www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/

[15] *Porovnejhosting.cz: Srovnávač webhostingů s největšími slevami* [online]. [cit. 2017-05-17]. Dostupné z: https://porovnejhosting.cz/webhosting

[16] *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems* [online]. [cit. 2017-05-17]. Dostupné z: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

[17] *Oracle: Top 10 Reasons to Choose MySQL for Web-based Applications* [online]. [cit. 2017-05-17]. Dostupné z: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

[18] *Number of apps available in leading app stores as of March 2017* [online]. [cit. 2017-05-17]. Dostupné z: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[19] *Apple Official Developer Support: Core OS Layer* [online]. [cit. 2017-05-17]. Dostupné z: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreOSLayer/CoreOSLayer.html#//apple_ref/doc/uid/TP40007898-CH11-SW1

[20] *Apple Official Developer Support: Core Services Layer* [online]. [cit. 2017-05-17]. Dostupné z: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreServicesLayer/CoreServicesLayer.html#//apple_ref/doc/uid/TP40007898-CH10-SW5

[21]
*Apple Official Developer Support: Media Layer* [online]. [cit. 2017-05-17]. Dostupné z: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/MediaLayer/MediaLayer.html#//apple_ref/doc/uid/TP40007898-CH9-SW4

[22] *Apple Official Developer Support: Cocoa Touch Layer* [online]. [cit. 2017-05-17]. Dostupné z: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html#//apple_ref/doc/uid/TP40007898-CH3-SW1

[23]
*Number of available applications in the Google Play Store from December 2009 to March 2017* [online]. [cit. 2017-05-17]. Dostupné z:

https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[24] *Android Deveolepers: Hardware Abstraction Layer (HAL)* [online]. [cit. 2017-05-17]. Dostupné z: https://developer.android.com/guide/platform/index.html#hal

[25] VINCENT, James. *99.6 percent of new smartphones run Android or iOS* [online]. [cit. 2017-05-17]. Dostupné z: https://www.theverge.com/2017/2/16/14634656/android-ios-market-share-blackberry-2016

[26] LANGER, Michal. *Podíl OS Android na světovém trhu dosáhl již téměř 88%, iOS je druhý a ostatním platformám se vůbec nedaří* [online]. [cit. 2017-05-17]. Dostupné z: http://androidmarket.cz/ruzne/podil-os-android-na-svetovem-trhu-dosahl-jiz-temer-88-ios-je-druhy-a-ostatnim-platformam-se-vubec-nedari/

[27] *Microsoft Developer Networm: Microsoft Application Architecture Guide, 2nd Edition* [online]. [cit. 2017-05-17]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff650706.aspx

[28] *Létající cirkus: Python* [online]. [cit. 2017-05-17]. Dostupné z: https://www.root.cz/clanky/letajici-cirkus/

[29] *Web Frameworks for Python: Python* [online]. [cit. 2017-05-17]. Dostupné z: https://wiki.python.org/moin/WebFrameworks

[30] JANKŮ, Dominik. *Ideální webový framework pro Python?* [online]. [cit. 2017-05-17]. Dostupné z: https://blog.root.cz/djanku/idealni-webovy-framework-pro-python/

[31] *Which is better to build web applications and why: PHP, Python or Ruby?* [online]. [cit. 2017-05-17]. Dostupné z: https://www.quora.com/Which-is-better-to-build-web-applications-and-why-PHP-Python-or-Ruby

[32] CARLA, Siya. *The Battle of Python v/s PHP – What's Better for Your Web Application Development? Read more at http://www.business2community.com/strategy/battle-python-vs-php-whats-better-web-application-development-01715981#cszVCw6POMe4bBRt.99* [online]. [cit. 2017-05-17].

[33] *The Transparent Language Popularity Index* [online]. [cit. 2017-05-17]. Dostupné z: http://lang-index.sourceforge.net/

[34] *CanvasJS: HTML5 JavaScript Charts* [online]. [cit. 2017-05-17]. Dostupné z: http://canvasjs.com/

[35] *Php: PHP Data Objects* [online]. [cit. 2017-05-17]. Dostupné z: http://php.net/manual/en/book.pdo.php

[36] *Android Developers: Android 4.2 APIs* [online]. [cit. 2017-05-17]. Dostupné z: https://developer.android.com/about/versions/android-4.2.html

[37] *Android Developers: AsyncTask* [online]. [cit. 2017-05-17]. Dostupné z: https://developer.android.com/reference/android/os/AsyncTask.html

# Seznam příloh



# Příloha na CD

### Webová aplikace

Soubory s webovou aplikací jsou v adresáři `htdocs.`

### Instalace

Pro korektní práci s aplikací a správné vykreslení všech prvků je nutné mít prohlížeč s podporou HTML5.

SQL dump pro vytvoření databáze a nahrání testovacího obsahu se nachází v kořenovém adresáři a je pojmenován jako `sql_dump.sql.`

### Mobilní aplikace

Projekt pro Android Studio je v adresáři `android`. Pro spuštění aplikace je vyžadována nejnovější verze Android Studia s nainstalovaným API 17.

### Instalace

Před instalací aplikace na mobilní zařízení je nutné upravit několik informací ve zdrojových souborech. Konkrétně se jedná o modul `BackgroundWorker.java`, na řádku 34. Ten obsahuje následující kód:

```
String login_url = "http://{ip_adresa}:{port}/sqlsam.php";
```

Obsah pole `ip_adresa` je nutné změnit na adresu, na které je spuštěn server, za dvojtečkou následuje číslo portu, na kterém je spuštěn server Apache.

Stejnou změnu je nutné provést i na řádku 48 modulu `MainActivity.java`

```
private static final String URL = "http://{ip_adresa}:{port}/user_control.php";
```

Na mobilním zařízení je nutné zapnout v nastavení vývojářský mód. Způsob, jakým se toto provede, se liší v závislosti na verzi a neexistuje obecný postup. Proto doporučuji vyhledat řešení na internetu.

Považuji za nutné zmínit, že mobilní aplikace nebyla testována na zařízením se starší verzí systému Android, než je 4.2. Proto nemohu zaručit funkčnost na nižších verzích OS.

Po výše uvedených úpravách je aplikace připravena k provozu.